

José Miguel Horcas José Ángel Galindo
Richard Comploi-Taupe Lidia Fuentes (Eds.)

ConfWS 2023
25th International Workshop on Configuration

Málaga, Spain, September 6-7, 2023
Proceedings

© 2023 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. Re-publication of material from this volume requires permission by the copyright owners.

Editors' addresses:

Universidad de Málaga
E.T.S. Ingeniería Informática
Bulevar Louis Pasteur, 35. Campus de Teatinos
29071 Málaga, Spain

horcas@uma.es, jagalindo@us.es, richard.taupe@siemens.com, lfuentes@uma.es

Preface

The 25th International Workshop on Configuration (ConfWS 2023) is a vibrant hub for researchers and industry professionals interested in configuration technology. The product configuration field refers to the arrangement, composition, or setup of various components or elements within a system, product, or solution. The need for configuration is wide-ranging beyond software in today's life.

We are proud that the Configuration Workshop (ConfWS 2023) is celebrating its 25th anniversary. This year, it is hosted by the University of Málaga in Spain as a stand-alone two-day event where high-quality research in all configuration-related technical areas is presented. The program includes special sessions about visualization and configuration, configuration tasks, constraint programming, configuration applications, answer set programming (ASP), and product configuration.

There were 18 papers submitted for peer review to ConfWS 2023. 16 papers were selected for publication in these proceedings after a review by three independent reviewers per paper, 15 as regular papers and 1 as a position statement paper. One more paper was accepted as a one-page abstract for presentation in the workshop (Lothar Hotz, Rainer Herzog, Yibo Wang, and Stephanie Von Riegen: “*HydrA - A Hybrid Architecture for Adaptive, Monitoring-based Bottom-up Configuration*”). An additional paper was invited to be presented as a keynote (Michel Aldanondo and Elise Vareilles: “*After 25 years in the product configuration field, some remaining topics that interest us*”).

The ConfWS 2023 introduced the figure of “Award Chair” in the organization committee, a role played by Michel Aldanondo (Université de Toulouse, France), who was in charge of managing the selection of the best paper awards in a two-phase audience vote at the end of the workshop. This forum presents awards to the best papers since ConfWS 2015. This year, in ConfWS 2023, the Best Paper Award winner was “*Visualization in Configurators: Reflections for Future Research*” by Enrico Sandrin and Cipriano Forza, and the Best Student Paper Award winner was “*Product Variant Master in the Construction Industry: A Synthesis of Construction Product Platforms*” by Irene Campo Gay and Lars Hvam.

We thank the authors for their submissions, the program committee for their hard work, the University of Málaga and the ITIS Software for supporting this event, the E.T.S. Ingeniería Informática for the space to host the event, and SIEMENS for sponsoring ConfWS 2023. The following projects by FEDER/Ministry of Science and Innovation/Junta de Andalucía/State Research Agency, and EU, also supported the workshop: *TASOVA PLUS* research network (RED2022-134337-T), *IRIS* (PID2021-122812OB-I00), *LEIA* (UMA18-FEDERJA-157), *Data-pl* (PID2022-138486OB-I00), *METAMORFOSIS* (FEDER_US-1381375), and *DAEMON* (H2020-101017109).

September 2023

José Miguel Horcas, José Ángel Galindo,
Richard Comploi-Taupe, Lidia Fuentes

Workshop Chairs

José Miguel Horcas, University of Málaga, Spain
José A. Galindo, University of Seville, Spain
Richard Comploi-Taupe, Siemens, Austria
Lidia Fuentes Fernández, University of Málaga, Spain

Award Chair

Michel Aldanondo, Université de Toulouse - IMT Mines Albi, CGI Albi, France

Program Committee

Lothar Hotz, Hamburger Informatik Technologie-Center, Germany
Ángel Jesús Varela Vaca, University of Seville, Spain
Abdourahim Sylla, Université Grenoble Alpes, France
Andreas Falkner, Siemens, Austria
Elise Vareilles, ISAE SUPAERO Toulouse, France
Yue Wang, Hang Seng University, Hong Kong
Gerhard Friedrich, Alpen-Adria-Universität Klagenfurt, Austria
Alexander Felfernig, Graz University of Technology, Austria
Albert Haag, Product Management GmbH, Germany
Lars Hvam, Technical University of Denmark, Denmark
Sara Shafiee, Technical University of Denmark, Denmark
Franz Wotawa, Graz University of Technology, Austria
David Benavides, University of Seville, Spain
Tomas Axling, Tacton, Sweden
Tomi Mänistö, University of Helsinki, Finland
Jean-Guillaume Fages, Cosling, France
Enrico Sandrin, University of Padova, Italy
Thorsten Krebs, Encoway, Germany
Chiara Grossi, Sant'Anna School of Advanced Studies-Pisa, Health Science Research Center, Italy
Alois Haselboeck, Siemens, Austria
Markus Stumptner, University of South Australia, Australia
Mónica Pinto, University of Málaga, Spain
Inmaculada Ayala, University of Málaga, Spain

Volunteers

Laura Panizo, University of Málaga, Spain
María Fernández Márquez, University of Málaga, Spain

Contents

Visualization in Configurators: Reflections for Future Research <i>Enrico Sandrin, Cipriano Forza</i>	8
User Interface Expert for Configurators <i>Enrico Sandrin, Gerhard Leitner, Cipriano Forza</i>	12
Specifying Configurable Videos with Feature Models <i>Sebastian Lubos, Alexander Felfernig, Viet-Man Le</i>	22
Solving Constraint Satisfaction Problems with Database Queries: An Overview <i>Alexander Felfernig, Viet-Man Le, Albert Haag, Sebastian Lubos</i>	29
Game-based Configuration Task Learning with ConGuess: An Initial Empirical Analysis <i>Andreas Hofbauer, Alexander Felfernig</i>	34
Collaborative Recommendation of Search Heuristics For Constraint Solvers <i>Damian Garber, Tamim Burgstaller, Alexander Felfernig, Viet-Man Le, Sebastian Lubos, Trang Tran, Seda Polat-Erdeniz</i>	38
Solving Multi-Configuration Problems: A Performance Analysis with Choco Solver <i>Benjamin Ritz, Alexander Felfernig, Viet-Man Le, Sebastian Lubos</i>	45
Decision Heuristics in a Constraint-based Product Configurator <i>Matthias Gorenflo, Tomáš Balyo, Markus Iser, Tobias Ostertag</i>	51
Identifying Potential Applications of Service Configuration Systems in a Logistics Company <i>Erika Marie Strøm, Tine Meidahl Münsberg, Lars Hvam</i>	60
Multi-level configuration in smart governance systems <i>Salvador Muñoz-Hermoso, David Benavides, Francisco Jose Dominguez Mayo</i>	67

Dynamic Aggregates in Expressive ASP Heuristics for Configuration Problems	
<i>Richard Comploi-Taupe, Gerhard Friedrich, Tilman Niestroj</i>	75
Towards a formalization of configuration problems for ASP-based reasoning:	
Preliminary report	
<i>Nicolas Röhling, Torsten Schaub, Tobias Stolzmann</i>	85
Interactive Configuration with ASP Multi-Shot Solving	
<i>Richard Comploi-Taupe, Andreas Falkner, Susana Hahn, Torsten Schaub, Gottfried Schenner</i>	95
PERFECT: PErformant and Robust read-to-fly FIEet ConfiguraTion: from robot to mission plan	
<i>Elise Vareilles, Stéphanie Roussel, Gauthier Picard</i>	104
Construction of Decision Diagrams for Product Configuration	
<i>Maxim Popov, Tomáš Balyo, Markus Iser, Tobias Ostertag</i>	108
Product Variant Master in the Construction Industry: A Synthesis of Construction Product Platforms	
<i>Irene Campo Gay, Lars Hvam</i>	118

Visualization in Configurators: Reflections for Future Research

Enrico Sandrin¹ and Cipriano Forza¹

¹ University of Padova, Stradella San Nicola 3, 36100 Vicenza, Italy

Abstract

The increasing attention and investments in augmented reality (AR), virtual reality (VR), and mixed reality (MR) further highlight the importance of graphic representations as communication tools. However, numerous online configurators lack advanced visualization and very few utilize virtual reality. Considering the expense associated with advanced visualizations, it becomes crucial to understand the incremental utility of such visualizations within the configuration process. This positioning paper aims to call for and pave the way towards a deeper understanding of the role and value of visualization in configurators, not limiting to AR, VR, and MR but considering all forms of visualization.

Keywords

Configurator, product visualization, virtual reality, value

1. Introduction

"A picture is worth a thousand words!" This statement resonates with many of us, reflecting the widely recognized power of visual representations in effectively conveying concepts. For example, in the communication of a product for sale, it is highly beneficial to have effective and realistic visualizations of the product and its features [1]. The increasing investments in augmented reality (AR), virtual reality (VR), and mixed reality (MR) [2] further confirm the importance of graphic representations [3].

In the context of product configurators, a primary objective is to provide clear and easy-to-understand information about the choices, their impact on the overall product, and the resulting final product [4]. Many configurators employ visualizations of product parts or the entire product to help customers make informed choices [5]. However, numerous online configurators lack advanced visualization techniques and even fewer utilize virtual reality. Interestingly, many configurators without advanced visualization techniques still perform well in achieving their purpose. Therefore, there is no clear dominant visualization that companies can refer to.

Considering the differences in costs associated with the various product visualizations, it would be highly beneficial to know the benefits of the various product visualization modes in the different contexts, as well as the related implementation costs and challenges. Unfortunately, we are far from this ideal knowledge, with the consequent problem of limited

support provided to practitioners in making choices about product visualization in configurators.

This positioning paper aims to stimulate a scientific discussion to address the above-mentioned problem within the configurator development and usage community. Through our reflections, we aim to call for and pave the way towards a deeper understanding of the role and value of product visualization in configurators. When it will be available, this understanding will support companies in choosing visualizations for their configurators.

This discussion is particularly opportune now, given the growing attention to and availability of advanced solutions for visualizing configured products [e.g., 6, 7]. We suggest framing this discussion considering that the availability of powerful visualizations does not imply their profitability in all contexts and that, consequently, a company needs to assess them within its specific context and for specific purposes.

2. The importance of visualization in product customization

Product visualization is gaining importance in product customization. As product configuration takes place within the broader context of product customization, we begin our reflection by considering the following current trends.

First, customers, especially those who purchase custom products, demand more visual support. Visual

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

 enrico.sandrin@unipd.it (E. Sandrin); cipriano.forza@unipd.it (C. Forza)

 0000-0001-9170-0683 (E. Sandrin); 0000-0003-4583-2962 (C. Forza)



© 2023 Copyright for this paper by its authors. The use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). CEUR Workshop Proceedings (CEUR-WS.org)

experiences guide people's judgement, decision, ability to learn and retain information (90% of what we process is visual; we respond 60,000 times faster to imagery than text; 65% of the world is composed of visual learners) [3]. In today's market, potential customers have become increasingly accustomed to the visualization of products. Di et al. [1] provided evidence that images play a significant role in increasing buyer attention, trust, and conversion rates. Specifically, their research suggests that increasing the number of product images, which enhances the overall visual representation of the product, effectively improves sell-through. This result highlights the importance of providing a comprehensive visual experience to potential buyers to drive better sales outcomes [1]. This change in consumer behavior highlights the importance of visualization in helping customers customize products according to their individual needs [8]. During the customization process, it is crucial for customers to have a clear understanding of how the characteristics of the product align with their preferences to minimize any potential regret associated with their purchase decision [4]. By providing visual representations and interactive tools, companies can enable customers to make informed choices and ensure that the customized product meets their expectations. Effective visualization not only enhances the overall customization experience, but also mitigates concerns of post-purchase regret.

Second, companies have perceived this need and feel the need to invest in this direction. Big companies and brands that customize products such as Volkswagen, Nike, Ray-Ban with their investments in product visualization and with their interest in advanced product visualization project witness that they have perceived the customer need of seeing visually their products. But also SMEs that offer personalized products are perceiving the need to improve their product visualization [9].

Third, visualization technologies are making huge progresses. Product visualization technologies (AR, VR, etc.) are making incredible improvements [10]. For example, the launch of Apple Vision Pro, which tracks a person's eye movements and responds accordingly, and its integration with the Mac world, offers vast cutting-edge possibilities for users while being accessible.

Fourth, AR, VR, and MR are expected to have a fast growth in shopping. In 2028, the AR, VR and MR market will be nine times more than in 2021 [2]. Consumer confidence rises by 4% globally when using immersive technologies [11]. 71% of shoppers think they would shop more often if they used AR apps [12]. 61% of shoppers said they prefer to choose stores with AR over those without it [12]. 55% of shoppers said AR makes shopping more fun and exciting [12]. 83% of shoppers point to product images as the most influential factor in their purchasing decision [13]. The use of VR to build virtual online change rooms can help retailers improve conversion rates by more than 6.4%, increase order value by 1.6%, reduce fulfillment costs by 5%, and lower returns by 5.2% [14]. Virtual presence is something that is becoming increasingly important in online activities [15].

3. Virtual reality and configurators

The previous section justifies the relevance of visualization and, in particular, visualization advancements (such as VR, AR, MR) in product customization. Given the importance of visualization for product customization and configurators for mass customization, we would expect a lot of research on the use of visualization in configurators. Vice versa, current research on product visualization in the configuration process is limited, with a focus on advanced visualizations while neglecting the more commonly used basic forms of visualization.

The literature on product configuration has been paying attention to VR/MR/AR since 2003, and a great increase in attention started more recently in 2015. More specifically, by using the SCOPUS database and looking at this literature in five-year intervals, we can find the following trend with a five-year time frame: 1 publication (2000-2004), 3 publications (2005-2009), 2 publications (2010-2014), 8 publications (2015-2019), and 5 publications (2020-2023).

Most of this literature investigates the possibility of applying advanced visualization by developing and demonstrating feasibility of new applications and approaches [16-25]. Some publications, still focused on the feasibility, go further by considering capabilities, potentials, and usefulness of advanced visualization technologies in configurators [e.g., 26, 27].

Finally, five publications (mostly recent) focus more on investigating the benefits of adopting advanced visualization technologies than showing the possibility of using them [6, 7, 28-30]. To this last set of papers, we should add Hvam and Ladeby [5] that even do not focus on the benefits, consider different possibilities of advanced visualization and call for considering the relative difference in benefits while designing a visual configurator.

Surprisingly, no articles on the level of adoption of various visualization solutions and technologies are provided. Limited discussion of different benefits of different visualization solutions is provided. No comparison of different investment requirements and implementation difficulties is provided. Additionally, consideration of suitability for different company sizes and skills is limited.

4. Visualization in actual configurators

The presence of different product visualization modes for configurators is recognized [e.g., 5]. Unfortunately, a systematic and comparative characterization of these modes is not available. Below we recall and briefly describe a number of these modes. We rely on our knowledge, accumulated over years through the analysis of hundreds of online sales configurators, on a recent survey on the presence of configurator functionalities in almost 100 Italian and Austrian SMEs, and on the working with companies.

Augmented reality allows one to see the product in a real context of use, in a real world environment, in a

room, etc., for example, furniture in the customer's living room.

Virtual try-on allows one to see the product on the user's face or body. Examples of products that can use this technology include makeup, personal accessories, clothing, and shoes.

Virtual reality allows one to interact with and experience virtual products by creating a fully immersive virtual environment. Users can examine the product from different angles, change configurations, and assess its features, all within a simulated environment. Examples can be virtual showrooms of custom cars and interior design configurators.

3D walk-through allows one to virtually visit an environment (e.g. an apartment).

3D models allow one to view products from different angles and perspectives (e.g., footwear, jewelry, cars, furniture).

360 view allows one to explore a product from all angles. Users can interactively rotate the view horizontally and vertically to see the product from different perspectives.

2D image represents the product in two-dimensional graphical representation. It can be a drawing or a sketch.

Other modalities are *video and animations* of the real product or of the virtual image of the product, *photo* of the real product or *virtual image* of the product (e.g., rendering), *cross-section views* to see the internal structure or components.

Additionally, many techniques exist to interact with the configurator and see the effects of the user customization actions on the customized product. For example, *interactive configuration* allows users to manipulate certain product attributes directly on the image itself (e.g., users can click on different parts of a product image to change colors or select additional features). *Live previews* allow one to dynamically update the visual representation of the product depending on the user selections. The product visualization can *change simultaneously* with the modification of the chosen options or not. The product can *be set in motion* or not. The product visualization is *done only at the end* of the user configuration process or can also be *done during* the choice selection process.

5. Opportunities for future research and conclusions

The information and reflections presented up to this point have highlighted that: (1) The ability to offer suitable visual information to customers is increasingly important, even more when products are customized, eventually using an online sales configurator. (2) Many product visualization modes can be applied in configurators, both innovative (e.g., AR, VR, and MR) and established ones (e.g., 2D images and 3D models). (3) The business needs to which the visualization should respond are various (e.g., providing an approximate idea or a very detailed and realistic description of a product). (4) The business contexts in which to apply the visualization modes can be very different depending on many variables (e.g., customers, company size). (5) Studies investigating

the suitability of the various visualization modes for the various business needs and contexts are lacking.

Therefore, an important research opportunity consists of providing a comprehensive and comparative description of the various product visualization modes in configurators and investigating the effectiveness and challenges of different visualization modes across business needs and contexts. By describing and comparing the various forms of product visualization, as well as their utility and challenges in different contexts, research could help companies make well-informed choices regarding the adoption of visualization technologies in their configuration processes.

Let us conclude with a final consideration on the positioning of this line of research. Although exploring the new possibilities offered by virtual, mixed and advanced reality is crucial for technological advancements, it is equally important to provide managerial guidance to companies to make informed decisions regarding product visualization in configurators. Assessing the adoption levels of different technological solutions and understanding how users appreciate these solutions can provide valuable insights to companies. This information is not only valuable to companies but also helps researchers understand which aspects of technologies meet or fail to meet the needs of companies.

Acknowledgements

The authors thank (1) Farzaneh Bagheri Shahzadeh Aliakbari for her insights on the importance of visualization in customization focusing on the use of VR, AR, and MR; (2) Andreas Falkner for discussions on the relevance of this issue for companies; and (3) Universiy of Padova for the financial support Projects BIRD217418, DOR2271783, and DOR2334949.

References

- [1] W. Di, N. Sundaresan, R. Piramuthu, A. Bhardwaj, Is a picture really worth a thousand words? - on the role of images in e-commerce, Proceedings of the 7th ACM international conference on Web search and data mining, New York, New York, USA, Association for Computing Machinery, 2014, pp. 633–642. doi: 10.1145/2556195.2556226.
- [2] SkyQuest Technology, Global AR/VR/MR market, 2022. URL: <https://www.skyquestt.com/report/ar-vr-mr-market>.
- [3] ThreeKit, The visual economy, 2023. URL: <https://www.threekit.com/ebook/the-visual-economy#part1>.
- [4] A. Trentin, E. Perin, C. Forza, Sales configurator capabilities to avoid the product variety paradox: construct development and validation, Computers in Industry 64 (2013) 436-447. doi: 10.1016/j.compind.2013.02.006.
- [5] L. Hvam, K. Ladeby, An approach for the development of visual configuration systems, Computers & Industrial Engineering 53 (2007) 401-419. doi: 10.1016/j.cie.2007.05.004.

- [6] F. Turner, I. Welch, The mixed reality toolkit as the next step in the mass customization co-design experience, *International Journal of Industrial Engineering and Management* 10 (2019) 191.
- [7] Y. Lin, S. Yu, P. Zheng, L. Qiu, Y. Wang, X. Xu, VR-based product personalization process for smart products, *Procedia Manufacturing* 11 (2017) 1568-1576. doi: 10.1016/j.promfg.2017.07.297.
- [8] Srushti, Why should you show your product before it is manufactured! The power and importance of product visualization, 2017. URL: <https://srushtiviz.com/blog/why-should-you-show-your-product-before-it-is-manufactured-the-power-and-importance-of-product-visualization/>.
- [9] S. Suzic, E. Sandrin, N. Suzic, C. Forza, A. Trentin, Product configuration activities in SMEs and their digitalization: preliminary results of a survey study, in C. Forza, L. Hvam, A. Felfernig (Eds.), *Proceedings of the 22nd International Configuration Workshop*, September 10-11, Vicenza, Italy, 2020, pp. 106-113.
- [10] C. Flavián, S. Ibáñez-Sánchez, C. Orús, The impact of virtual, augmented and mixed reality technologies on the customer experience, *Journal of Business Research* 100 (2019) 547-560. doi: 10.1016/j.jbusres.2018.10.050.
- [11] Accenture, Try it. Trust it. Buy it, 2020.
- [12] H. Ebbesen, C. Machholdt, Digital reality changes everything: step into the future, Deloitte Development LLC (2019).
- [13] D. Ward, Product visuals for Magento with Threkit, 2020.
URL: <https://www.threkit.com/blog/product-visuals-for-magento-with-threkit>.
- [14] A. Startup, 50+ v-commerce statistics you need to know in 2023, 2023.
URL: <https://www.stylight.com/insights/news/50-v-commerce-statistics-you-need-to-know/>.
- [15] L. D. Hollebeek, M. K. Clark, T. W. Andreassen, V. Sigurdsson, D. Smith, Virtual reality through the customer journey: framework and propositions, *Journal of Retailing and Consumer Services* 55 (2020) 1-12. doi: 10.1016/j.jretconser.2020.102056.
- [16] M. Mengoni, D. Raponi, R. Raffaeli, A web-enabled configuration system for interior design, *Computer-Aided Design and Applications* 12 (2015) 753-764. doi: 10.1080/16864360.2015.1033341.
- [17] M. Mondellini, S. Arlati, S. Mottura, V. Colombo, E. Biffi, A. Davalli, M. Sacco, A usability study of an application to configure virtual reality training environments for wheelchair users, *Computer-Aided Design and Applications* 20 (2023) 134-144. doi: 10.14733/cadaps.2023.S6.134-144.
- [18] L. Potseluyko, F. Pour Rahimian, N. Dawood, F. Elghaish, A. Hajirasouli, Game-like interactive environment using BIM-based virtual reality for the timber frame self-build housing sector, *Automation in Construction* 142 (2022) 1-18. doi: 10.1016/j.autcon.2022.104496.
- [19] A. Romani, M. Levi, Parametric design for online user customization of 3D printed assistive technology for rheumatic diseases, in L. T. De Paolis, P. Bourdot (Eds.), *Augmented Reality, Virtual Reality, and Computer Graphics*, Springer, Cham, Switzerland, 2020, pp. 174-182. doi: 10.1007/978-3-030-58468-9_14.
- [20] P. Novak, P. Kadera, M. Wimmer, Model-based engineering and virtual commissioning of cyber-physical manufacturing systems — Transportation system case study, *Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, September 12-15, Limassol, Cyprus, IEEE, 2017, pp. 1-4. doi: 10.1109/ETFA.2017.8247743.
- [21] M. Gebert, W. Steger, R. Stelzer, K. Bertelmann, Meta-model for VR-based design reviews, *Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 4: Design Methods and Tools*, , August 21-25, Vancouver, Canada, 2017, pp. 337-346.
- [22] A. Bachvarov, S. Maleshkov, P. Häfner, J. Katicic, Design-by-the-customer through virtual reality, *Proceedings of the 4th International Conference on Advanced Research and Rapid Prototyping*, Leiria, Portugal, 2009, pp. 561-566.
- [23] C. Calderon, M. Cavazza, D. Diaz, A new approach to virtual design for spatial configuration problems, *Proceedings of the 7th International Conference on Information Visualization*, London, UK, 18-18 July 2003, IEEE, 2003, pp. 518-523. doi: 10.1109/IV.2003.1218034.
- [24] C. Calderón, M. Cavazza, D. Diaz, CLP a technology for the interactive resolution of spatial configuration tasks in a virtual environment, *Journal of Information Technology in Construction* 11 (2006) 325-341.
- [25] I. Graessler, P. Taplick, Supporting creativity with virtual reality technology, *Proceedings of the Design Society: International Conference on Engineering Design* 1 (2019) 2011-2020. doi: 10.1017/dsi.2019.207.
- [26] Y. Liu, Y. Zhang, S. Zuo, W.-T. Fu, BoatAR: a multi-user augmented-reality platform for boat, *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, November 28 - December 1, Tokyo, Japan, Association for Computing Machinery, 2018, pp. 1-2, Article 74. doi: 10.1145/3281505.3283392.
- [27] F. Górski, P. Buń, R. Wichniarek, P. Zawadzki, A. Hamrol, Immersive city bus configuration system for marketing and sales education, *Procedia Computer Science* 75 (2015) 137-146. doi: 10.1016/j.procs.2015.12.230.
- [28] M. Fiorentino, M. Ricci, A. Evangelista, V. M. Manghisi, A. E. Uva, A multi-sensory in-store virtual reality customer journey for retailing: a field study in a furniture flagship store, *Future Internet* 14 (2022) 1-12. doi: 10.3390/fi14120381.
- [29] M. Simoni, A. Sorrentino, D. Leone, A. Caporuscio, Boosting the pre-purchase experience through virtual reality. Insights from the cruise industry, *Journal of Hospitality and Tourism Technology* 13 (2022) 140-156. doi: 10.1108/JHTT-09-2020-0243.
- [30] R. Rolland, E. Yvain, O. Christmann, E. Loup-Escande, S. Richir, E-commerce and web 3D for involving the customer in the design process: the case of a gates 3D configurator, *Proceedings of the Virtual Reality International Conference*, March 28-30, Laval, France, Association for Computing Machinery, 2012, pp. 1-8, Article 25. doi: 10.1145/2331714.2331743.

User Interface Expert for Configurators

Enrico Sandrin¹, Gerhard Leitner² and Cipriano Forza¹

¹ University of Padova, Stradella San Nicola 3, 36100 Vicenza, Italy

² University of Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria

Abstract

A user interface (UI) expert for configurators is a specialist who focuses on designing and optimizing the interfaces of configurator systems. Configurators are tools that allow users to customize or personalize products or services according to their specific preferences and requirements. The role of a UI expert in enabling users to perform such customization processes is therefore crucial in today's digital landscape. As technology continues to advance and user expectations grow, user interfaces become an essential component of any product or service, with regard to the product/service itself or in its selling. A UI expert provides specialized knowledge and skills to create intuitive and user-friendly interfaces that enhance the overall user experience. Therefore, companies that want to implement or already use configurators benefit from understanding and applying the specific competencies of UI experts for configurators. The present paper sheds light on the tasks, individual competencies, and training requirements of the UI expert for configurators in an industrial context. In addition, the profile is also compared with European standard descriptors.

Keywords

Mass customization, configuration, configurator, user interface expert, professional profile, individual competencies

1. Introduction

Configurators can be observed as an interface between companies and their customers. They allow the customization or personalization of products or services according to the customers' specific preferences and requirements. The user interface (UI) plays a critical role in this context, because, as pointed out in Carroll [1]: "The interface connects the technical system and the user, and it therefore has a potentially big impact on the success and failure of the *human-machine system as a whole*" [1: p. 55]. Well-designed UIs are of specific importance in product configurators [2], as they are increasingly used by companies to initially consult clients to save scarce human resources. Challenges related to bad UIs are frequently reported [e.g. 3, 4]. For example, an un-user-friendly interface is the most cited cause of difficulties in using product configurators in the survey of Zhang and Helo [3] and lack of usability is among the worst defects of online configurators in the survey of Leclercq et al. [4]. Therefore, the role of an expert responsible for the UI of configurators has become essential in recent years because of the spread of configurators guiding customers and supporting company personnel through the customization process (besides desktop environments also on mobile devices, digital kiosk systems, etc.), in the product/service itself or in its

selling. A central task of UI experts is to bridge the gap between technology and users [5]. They understand the needs, behaviors, and preferences of the target audience and translate them into design decisions. This goes beyond the characteristics of the interface alone and covers all relevant aspects of user experience (UX) [6]. By, for example, conducting user research, applying user-centered design processes (UCD), and related methods (e.g. prototyping, usability testing) [7], UI experts ensure that the interface meets the users' requirements and expectations and fits in the context of use. UI experts bring specialized knowledge and skills to create intuitive and easy-to-use interfaces, for example, by making it easier for users to navigate and interact with the system on different digital platforms (desktop, web, mobile). By taking into account related standards, guidelines, and interaction patterns it is possible to flatten the learning curve, improve efficiency, and minimize errors, all of them resulting in increased user satisfaction. A positive UX leads to greater user engagement, customer loyalty, and ultimately business growth [8, 9].

User interface experts for configurators pay specific attention to the mechanisms of configuration, for instance, the complexities involved in configuring products and translate that understanding into an interface that supports the specific needs of customers in this context (e.g. enhanced navigation aids in the form of wizards, corresponding overview / detail

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

 enrico.sandrin@unipd.it (E. Sandrin); gerhard.leitner@aau.at (G. Leitner); cipriano.forza@unipd.it (C. Forza)

 0000-0001-9170-0683 (E. Sandrin); 0000-0002-3084-0727 (G. Leitner); 0000-0003-4583-2962 (C. Forza)

 © 2023 Copyright for this paper by its authors. The use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

visualizations, etc.). The professional portfolio of the UI expert involves layout definition activities, graphic and dialog design, and customization activities in product configuration systems [10]. In addition, it includes the evaluation of user interaction and experience and derives suggestions for improvements. Thus, the UI expert for configurators analyzes all important aspects of human-machine interaction, as well as the users' perceptions of system characteristics, covered in the specifications of usability engineering [11] and UX [12]. Besides the skills related to the characteristics of the UI and interaction, the UI expert for configurators possesses the appropriate soft skills to work in an interdisciplinary team and to communicate with users, involve them in development processes, understand and satisfy their different requirements and needs. In addition to knowledge and skills in ICT, they must understand human aspects that can influence interaction with a configuration system.

Unfortunately, notwithstanding the importance of the UI expert for configurators and their peculiarities in terms of knowledge and skills required, there is no discussion undergoing in the configurator community about this professional profile and how it could be developed in different organizational contexts and through training curricula. The Configuration Workshop is an appropriate place to start a joint academic and practitioner discourse on this professional figure. This is because this is a discussion place where the discussion on UIs for configurator has been on the table for several years [e.g. 6, 13, 14, 15]. This discussion should spread to other places where the characteristics and importance of the UI for configurators has been discussed [e.g. 2, 15, 16-18].

The objective of this paper is to start a scientific discussion on the professional profile of an UI expert for configurators in an industrial context. This paper provides a description of the activities and individual competencies that refer to this professional figure, as well as the contents of an educational concept and training activities required to develop such a professional figure.

The idea presented is based on the personal experiences of the authors in consulting and training, as well as on the experiences of third parties and the information collected by the authors over more than 20 years of activity with configurators in Austria and Italy. This knowledge has been integrated with academic literature and with the descriptions of professional figures taken from recognized frameworks. The results have been discussed in-depth with seven companies (mostly with entrepreneurs), one training and transfer center, and one industrial association of small and medium-sized enterprises (SMEs).

2. Theoretical background

In today's digitalized world, user interfaces play a critical role as they play an essential role in building a bridge between technology and users [5]. The role of UI experts is to ensure that these interfaces meet user requirements and expectations, thus improving aspects of usability, such as making it easier for users

to navigate and interact with the system [5, 19]. In particular, UI experts play a vital role in creating interfaces that help users solve complex tasks, such as in the context of configurable products. The central challenge in this regard is to create interfaces that are aesthetically pleasing despite the complexity of the basic tasks they support. According to related models [cf. e.g. 20], UI experts therefore have to have a deep understanding of general interface design principles and specific requirements of different platforms (e.g. desktop, mobile) as well as of psychological aspects related to, for example, color theory, typography, and visual hierarchy. They carefully design layouts, evaluate alignments, spacing, and the selection of visual elements to create an optimal UX based on an interface that is visually appealing and consistent with a corporate's visual and overall identity. A visually pleasing interface can evoke positive emotions, establish trust, and enhance the overall perception of the product or service. To achieve these results, the UI expert has to collaborate intensively with different departments and employees of a company, but specifically with the other members of the development team, such as UX experts, developers, and product managers. The team cooperatively ensures that the UI design is aligned with overall UX requirements, the company-specific strategy, and technical feasibility. The knowledge of UI experts in usability and interaction design helps make informed decisions on different levels, for example, regarding the placement of elements or the utilization of interaction patterns such as responsive design across different devices and platforms.

To be able to assume the described role, the UI expert for configurators typically has a background as a professional interface designer who specializes in configurator environments, responsible for creating UIs of software systems using front-end prototyping tools or integrated development environments (IDEs).

Product configurators represent a special category of human-computer systems and therefore require a deeper understanding and knowledge beyond standard UI development knowledge. Configurators are software applications that not only provide the selection of existing standard products but also support the customization of services and products and the creation of new product variants. Application areas are, for example, sales configuration processes and/or technical configuration processes [e.g. 21, 22]. In the case of sales configuration, product configurators assist a potential customer or a salesperson who interacts with the customer to fully and correctly specify a product solution among all possible solutions offered by the company. In the case of technical configuration, product configurators support the creation of a technical documentation that a company uses to create a desired customer solution.

The market offers a significant number of software packages that allow companies to create their own sales and technical configuration solutions. Alternatively, a company can create its own product configurator from scratch. In either case, the design of UIs that enable interaction between human users and an ICT system in the background is crucial. In contrast to other types of UI, configurators usually consist of a more complex interface architecture because there is a

need to support different actors in different parts of a configuration process. Potential users of these interfaces can be, for example, product specialists, sales professionals, knowledge engineers, designers, installers, system administrators, or end customers.

User interfaces of sales configurators supporting these different types of user, therefore, play a specific role in this context, specifically in the direction of end customers [4]. In contrast to the interfaces addressing experts, end-user interfaces should adequately convey the configuration possibilities without overwhelming the customer, thus significantly influencing the customer's perceived benefits when configuring customized products. The related literature identifies five configurator interface capabilities that are extremely important in improving the customers' perceived benefits of customized products and customization experiences [23-27]: (1) *focused navigation* (to quickly focus a potential customer's search on the solutions in the product space of a company that are most relevant to the customer); (2) *flexible navigation* (to enable users to easily and quickly modify a product configuration they have previously created or are currently creating); (3) *benefit-cost communication* (to effectively communicate the consequences of configuration decisions made by a potential customer in terms of advantages and disadvantages); (4) *user-friendly product space description* (to adapt a company's product space description to the individual characteristics of a potential customer and the situational characteristics of his or her use of the sales configurator); (5) *easy comparison* (to help users compare previously created product configurations).

These sales configurator capabilities increase the benefits perceived by customers when they configure customized products using the sales configurator (benefits of *utilitarian*, *uniqueness*, *self-expressiveness*, *hedonic*, and *creative-achievement*) [24, 26]. Since these capabilities, at least some of them, have a synergistic effect on the benefits perceived by the customer [23], it is important to balance them in the design of a configurator UI.

More recently, customers' need for social interaction during configuration activity has also been recognized as an important aspect to consider when designing a sales configurator [28]. This additionally supports the need of a broad range of related skills emphasized before, which an UI expert for configurators has to have. In addition to technical computer knowledge about software programming, e-commerce, websites, and other applications and tools that interact with the configurator (e.g. social media, chatbots, artificial intelligence-related applications, 3D graphics), this expert must have know-how in marketing, sales, human factors, social and psychological aspects that may influence the effectiveness of the product configurator.

Another peculiar characteristic of configurator UIs is that there are many different solutions for creating configurators' UIs [e.g. 29, 30], therefore, is difficult to identify guidelines [29, 31] and effective unique standards for creating such UIs [29, 32, 33]. Moreover, configurator UIs can change rapidly even from year to year [30]. All these characteristics require to be able to identify the most effective UI design for a particular

configuration context and to be ready to continuous redesign and updating of the UI.

In summary, the UI expert for configurators must be able to interact with, understand, and satisfy different types of stakeholders in a development process, specifically users with different requirements, backgrounds, and expectations [4, 34-36].

3. Method

The main objective of this work is to gather and disseminate information on the topic mainly to players in the industrial sector, such as SMEs, who do not have the resources and possibilities to access academic literature and other sources of related information. In comparison to the academic field or large enterprises which have in-company resources to deal with the problem, the presented approach addresses the dearth of prior research focusing on the role of UI expert for product configurators in specific industrial contexts. The empirical knowledge of this attempt is mainly based on the personal knowledge of the authors of this paper, complemented by a theoretical foundation based on the related academic literature.

The authors have between 15 and 24 years of experience in the fields of studying and/or guiding the implementation of configuration systems, collaborating and exchanging experiences with configurator experts, training individuals to design and implement configuration systems, and evaluating and guiding businesses in the eventual implementation of mass customization mechanisms using configuration systems. Additionally, they have researched the effects of online sales configurator capabilities on users' perceived benefits in using those configurators. The authors together developed deep insights into all of the pertinent problems in this area, thanks to the collection of expertise of executors, managers, consultants, trainers, and researchers on these themes. They worked with practitioners mainly in Italy and Austria, although they had multiple occasions to interact across several European countries. They have also conducted research on organizational design and individual competencies for mass customization that helped in identifying relevant activities of the role and the related individual competencies.

The original proposal for UI experts for configurators activities, their individual competencies, and proposed training activities was created jointly by two authors. Subsequently, this proposal underwent a number of revisions where each of the three authors suggested integration or changes. To be able to see the proposal with fresh eyes, we completed three distinct cycles of changes, spaced widely apart from each other. Two specialists, one working for a training and transfer center and the other for an SMEs association, have since conducted final checks.

Essential individual competencies were determined considering the processes and related activities and returning to the authors' experiences on that topic. Individual competencies were grouped into (i) knowledge, (ii) skills, and (iii) transversal skills and competences. These categories are aligned to European reference frameworks, such as ESCO

(European Skills, Competences, Qualifications and Occupations), where they have the same meaning as in the present paper. In this paper, the term knowledge - "is composed of the facts and figures, concepts, ideas and theories which are already established and support the understanding of a certain area or subject" [37: p. 7] and the skills "are defined as the ability and capacity to carry out processes and use the existing knowledge to achieve results" [37: p. 7]. Transversal skills and competences (TSCs) are understood as "learned and proven abilities which are commonly seen as necessary or valuable for effective action in virtually any kind of work, learning or life activity. They are considered-labelled 'transversal' because they are not exclusively related to any particular context (job, occupation, academic discipline, civic or community engagement, occupational sector, group of occupational sectors, etc.)" [38: p. 5].

The reason for choosing this approach is to allow starting a discussion not only among academics but also with training centers and practitioners. This categorization has been utilized to preserve a language widely used in the organizations where we operate while still preserving comparability to past research and international and European classifications. Finally, standard descriptors were used to characterize the figure of the UI expert for configurators. The use of standard descriptors aims to make it easier to recognize competencies across different regions and systems.

To strengthen the validity of our findings, the profile was tested with top managers and entrepreneurs of SMEs. Five of these assessments were supervised by non-academics working for an SME association, and two were guided by one of the authors. These validity checks allowed external control by informed persons who were not affiliated with the authors. The evaluations sought to determine whether the professional figure's description was understandable and meaningful, whether this figure is useful in companies that have to develop configurators and respective UIs, and whether the proposed training paths are relevant and flexible enough to adapt to different contexts.

4. Activities of the UI expert for configurators

The main activities of the UI expert for configurators can be grouped into four parts with the following focal points; (a) understanding of user needs, (b) development of information architecture, (c) development of wireframes and prototypes, and (d) detailed visual interface design.

Understanding of user needs covers the understanding of the user requirements and needs (applying user research and workflow analysis) in order to ensure usability and overall UX.

Development of information architecture deals with the organization of information in an understandable way, translating the requirements into a structure, for example, on the basis of style guides, design systems, design patterns, and attractive UI elements.

Development of wireframes and prototypes involves the initial development, the testing, and iterative further development and refinement of wireframes and prototypes.

Visual interface design consists of four subsets of activities. The first one is UI development planning covering the content to be conveyed, utilized controls, visual design aspects, branding, appropriate navigation aids, benefit-cost communication, user-friendly description of the product space, and the support of easy comparison. The second subset covers the proper application of visual principles: aesthetics in UI, gestalt principles (similarity, proximity, etc.), grouping and organization, hierarchy. The third one deals with the identification of design problems, taking into account the characteristics and presentation of the product from a business, technical, production, or management perspective, as well as the configurator's interdependencies with other business processes and systems, business requirements, and the development of workable solutions. The fourth subset involves stakeholders in the presentation of content and design solutions for configurators and incorporates their feedback into the new design.

5. Individual competencies of the UI expert for configurators

To help the reader to understand which specific individual competencies are needed, the presentation of these competencies of the UI expert for configurators is exposed by the fundamental activities. For each activity, we should have reported the transversal skills and competences too; however, since these skills and competences tend to be common across multiple activities, we reported them separately in Subsection 5.5.

5.1. Understanding user needs to improve usability and overall UX

Understanding user needs to improve usability and overall UX requires the ability to refer to user research from different disciplines and the ability to conduct empirical evaluations on requirements and actual ICT use. Specific knowledge and skills are needed to perform this task well.

Knowledge. The required knowledge includes: (a) interdisciplinary Design Sciences Research perspectives; (b) UX research methods, such as: usability tests; interviews; card sorting (for categorization and hierarchy); eye-tracking and click tests; multivariate and A/B testing; desirability studies; expert/heuristic reviews; surveys; diary studies (recording behaviors or thoughts); personas; participatory design workshops, focus groups, social media listening, interviews; (c) customer research on, for example, benefits, needs and requirements in the sales configuration process; (d) specific requirements of product configurator users (company-internal / external).

Skills. The required skills include: (a) designing a User Centered Design Process (UCD), for example, involving the activities of designing, planning, and conducting UX and usability studies; (b) defining and planning of accompanying activities of empirical social research, such as communication activities; (c) involvement of different stakeholders (company representatives, industrial partners, funding organizations, public authorities); (d) integrating end customers on the basis of an UCD-process, e.g. for requirements and needs elicitation or as participants of usability studies; (e) analyzing the results of usability studies and proposing findings; (f) communicating methodological approaches and results through engaging presentations; (g) initially developing and refining design solutions (wireframes / paper prototypes) based on research activities.

5.2. Information architecture

Information architecture requires one to allocate the contents and procedures of the configurator and organize them according patterns of hierarchy or taxonomy, defining the basic concepts for navigating the website.

Knowledge. The required knowledge includes: (a) organizational schemes and structures; (c) labeling systems; (d) navigation systems; (e) search systems; (f) platform capabilities and specifics (e.g. mobile [iOs, Android], desktop [Windows, MacOs, Linux]).

Skills. The required skills include: (a) analyze the configuration process and its dependencies on other business processes and systems; (b) organize content into taxonomies and hierarchies of information based on content strategy; (c) communicate conceptual overviews and overall website organization to the design team and clients; (d) research and design the fundamental concepts of website navigation; (e) define standards and specifications for the handling of semantic HTML markup, as well as for the format and handling of textual content; and (f) design and implementation of search optimization standards and strategies.

5.3. Development of wireframes and prototypes

Development of wireframes and prototypes requires developing, testing, and iterating to refine the wireframe.

Knowledge. The required knowledge includes: (a) characteristics and representation of the product from: a commercial point of view, a technical point of view, the production point of view, the management control's point of view; and (b) characteristics and representation of the production process.

Skills. The required skills include: (a) translating concepts into wireframes; (b) defining technical requirements; (c) developing creative ideas; (d) drawing design sketches; (e) translate requirements into a visual design; (f) create storyboards to generate ideas for solutions to user requirements; (g) propose and outline a set of visual concepts both on paper and using prototyping software applications (e.g. Figma,

Adobe XD); (h) create wireframes on paper and in digital format; (i) create paper prototypes to develop interactive designs; (j) design low- and high-fidelity prototypes; (k) design sketches of user flows.

5.4. Visual interface design

Visual interface design requires the development of the UI, the proper application of visual principles, the identification of design problems, and the participation of stakeholders in the presentation and incorporation of their feedback.

Knowledge. The required knowledge includes: (a) competitor solutions (standards) for the configuration process; (b) customer benefits, needs, and requirements for the sales configuration process; (c) user requirements for the product configurator; (d) internal and external user requirements for a configuration system; (e) configurator software programming; (f) web programming; (g) application usability; (h) graphic design editing software; (i) human-computer interaction foundations; (j) software UI design patterns; (k) software interaction design.

Skills. The required skills include: (a) work on cross-platform applications to develop UXs for mobile phones, tablets, and computers; (b) collaborate with other designers, product design and development teams, business analysts, engineers, and project managers; (c) collaborate regularly with clients to ensure that projects meet their requirements and key business objectives; (d) attend meetings to discuss and review project progress.

5.5. Transversal skills and competences

An UI expert for configurators also needs the following transversal skills and competences, which are not exclusively related to any particular activity but are useful in multiple areas of his or her work: customer orientation; problem solving; teamwork; working in a results-oriented group; attention to order and quality.

6. Description of the profile and reference areas with standard descriptors

Competences, knowledge, and skills related to the profile of the UI expert for configurators are found, organized by category, in ESCO (European Skills, Competences, Qualifications and Occupations) [39], the multilingual European classification of skills, competences and occupations. ESCO functions as a dictionary, describing, identifying, and classifying occupations and occupational skills relevant to the EU labor market, education, and training.

Standard descriptors that are most close to this new profile are identified in Table 1 in the Appendix. Table 1, reports the correspondence between the ESCO profiles, the NUP ISTAT profiles [40] and the areas of activities (ADA) of the "Atlante del Lavoro" (atlas of labor) [41] associated with these ISTAT profiles,

emphasizing the relationship with the UI expert for configurators. For the sake of completeness and precision, Table 1 contains hyperlinks to the ESCO standard terms used and the web pages where these terms are defined.

6.1. Description using ESCO profiles

The three ESCO profiles closest to the UI expert for configurators are the UI designer, the UI developer, and the UX analyst. The descriptions of these three profiles are given below.

User interface designer. Code: 2513.3. Description: User interface designers are in charge of designing user interfaces for applications and systems. They perform layout, graphics and dialogues design activities as well as adaptation activities.

User interface developer. Code: 2512.5. Description: User interface developers implement, code, document and maintain the interface of a software system by using front-end development technologies.

User experience analyst. Code: 2511.19. Description: User experience analysts assess client interaction and experience and analyze users' behaviors, attitudes, and emotions about the usage of a particular product, system or service. They make proposals for the improvement of the interface and usability of products, systems or services. In doing so, they take into consideration the practical, experiential, affective, meaningful and valuable aspects of human-computer interaction and product ownership, as well as the person's perceptions of system aspects such as utility, ease of use and efficiency, and user experience dynamics.

Other profiles that may have some similarities with the UI expert for configurators, though less than the first three, are the information and communications technology user support technicians, the webmaster, and the product and services manager.

7. Training activities

Training should be tailored depending on whether one or more companies are involved, whether knowledge of mass customization and product configurators is limited or advanced, whether a professional configurator is available or not, whether all possible interfaces are considered or only a subset, etc.

Training should also include learning in which the learner plays an active role, possibly both in evaluating and creating specific product configuration interfaces. This can obviously significantly increase the time required for training. It might be useful, especially for small companies, to analyze configuration websites, evaluate actual UIs, and create UIs.

From the above, a benchmark training course will last between 60 and 120 hours. Note that the minimum duration of the training course may be sufficient for companies that already have UX and UI design skills and need specialization in the specific features of a product configurator. Furthermore, the duration of the accompanying training course may also be significantly longer than the maximum reference duration if the involved persons do not have comprehensive UX and UI design skills.

For the profile of UI expert for configurator, the proposed training includes the following content.

7.1. Mass customization and configurational approach

The training on mass customization and the configurational approach includes: (a) variety, customization, and mass customization strategies; (b) degree of product customization; (c) configurational approach and efficient customization; (d) standard, configurable, and special products.

7.2. Product configuration and its digitization

The training on product configuration and its digitization includes: (a) activities in the product configuration process; (b) relationships between organizational context and configuration activities; (c) digitization and automation of the configuration process.

7.3. Product configuration systems

The training on product configuration systems includes: (a) architecture of configuration systems; (b) degree of automation of the configuration process; (c) IT solutions for the configuration process; (d) product models used in the configuration process; (e) configurators and connection/integration with other enterprise information systems (CRM, PDM / PLM, PIM, MPCS, Social Software, etc.).

7.4. Users of a configuration system and their needs

The training on characteristics of users of a configuration system and their needs includes: (a) perspectives of users and designers (conceptual models), overview of methods to involve users in a development process, such as usability testing, interviews, surveys; (b) different types of research to determine users' characteristics and needs: quantitative and qualitative, behavioral and attitudinal; (c) data analysis and result presentation.

7.5. Visual characteristics of configurator UIs

The visual characteristics of configurator UIs include, for example: aesthetics; gestalt principles; grouping and organization; hierarchy; grid and information density; typography and readability; icons; colors; illustration; presentation of data in configurators.

7.6. Designing the UX

Designing the UX includes analyzing UX aspects that go beyond actual interaction (e.g. platform preference

(iOS, Android), peer group identification, social network, etc.) based on the following activities: (a) evaluating user data; (b) creating personas; (c) working with scenarios and storyboards; (d) creating paper prototypes; (e) implementation planning and support.

7.7. Features and functionalities of configurator UIs

The features and functionalities of the configurator UI include: (a) how the features of sales configurators improve the customer's perceived value in configuring a custom product; (b) how the features of a technical configurator enhance the customer's understanding of technical feasibility (and limitations); (c) how specific desires and requirements that deviate from the standard can have a direct impact on pricing; and (d) how customers themselves can manipulate the price/functionality ratio of the product.

8. Discussion and conclusion

Individual competency research for employees working in mass customization situations is very limited ([42], [43], [44]). Despite the fact that previous work provides examples and considerations for a better understanding of the issue of individual competencies in a customization environment, they do not consider professional figures specifically designed for the mass customization context with the only, very recent, exception of the configuration manager [44]. In particular, previous studies do not consider a professional figure such as the UI expert for configurators, which is crucial when mass customization is realized with the use of configurators. This paper contributes to the effort of investigating individual competencies for mass customization by introducing the UI expert for configurators as a professional figure.

More precisely, we describe the activities in which this expert participates and has to perform in a leading position. We then gave suggestions for the knowledge, soft skills, and technical skills that this expert should possess. We compare the figure with current competency classification systems at the national and European levels, and we utilize these systems' standard descriptors to characterize the UI expert for configurators.

The evaluation of this figure with entrepreneurs, managers, a SMEs association, and a training and transfer center managers showed that all companies engaged in customization revealed a company's need for the listed individual competencies. This is an important empirical result. It shows that the need for these competencies is perceived by the target audience/industry. As a consequence, research on this professional figure would be welcomed by practitioners, since it corresponds to their needs.

Equally interesting is to see what the seven SMEs said when they contrasted our proposal with their specific organization design situation. A company recognized the presence of an employee with the competencies listed above, even though his position

was not specified with a name that reminds the UI expert for configurators. Other companies, less advanced in mass customization, thought that this professional figure includes so many competencies that it becomes very difficult to find adequate personnel immediately employable for an SME. However, they recognize the possibility to hire an external person with adequate knowledge and skills to make him/her productive in a reasonable amount of time in an SME environment. Other companies said that because of their small size it was difficult for them to get the required competencies from outside and that they have to develop them internally. Other companies said that they have two or more employees that together have the competencies covered by this figure. A company with a considerable amount of variety but not so high to justify the adoption of a configurator underlined that, in situations similar to theirs, a similar figure is needed, but without the specific knowledge of configurators. Hence, the practical world in the considered SMEs presents a highly differentiated situation with respect to building up of the needed competencies and their distribution across employees.

Therefore, our confrontation with seven SMEs, an industry association of SMEs and a training and transfer center provided a strong message, i.e. the essential need for the listed competencies, although their development and implementation pose a highly complex challenge, a challenge that depends on the context. This result constitutes an extensive opportunity and area for future research, which will be even more important in the future given the trend towards a greater digitalization.

With this paper, we started a conversation about the UI expert for configurators in businesses of various sizes, and we think this issue is relevant both scientifically and practically. To complement this paper, future studies might examine the effects of various training methods in developing certain professional figure competencies. Another research opportunity may be to examine how certain individual competencies improve a company's capability of mass customization or lessen the difficulties associated with configurator development.

Even though the activities and the competencies of the professional figure of the UI expert for configurators have been exposed to external scrutiny of industry experts and entrepreneurs, this scrutiny has to be considered as a preliminary one. More extensive scrutiny is needed to strengthen the results obtained and to link them to different company contexts and company performances. We have seen how the company size availability of human resources may influence the development of such a figure or split its competence across different employees. Further empirical evidence based on case studies as well as on surveys would be beneficial. Finally, lab experiments could be conducted to assess different teaching strategies to build the identified competencies.

Acknowledgements

The authors acknowledge financial support from the MC 4.0 Interreg V-A Italia–Austria project, Project ID: ITAT 1057. We would like to thank our MC 4.0 project

colleagues and partners, particularly Elena Fassa, Enrico Bressan, and Alessio Trentin, for their critical and constructive help.

References

- [1] J. M. Carroll (Ed.), HCI models, theories, and frameworks: toward a multidisciplinary science, Morgan Kaufmann Publishers, Elsevier Science, San Francisco, CA, 2003.
- [2] G. Leitner, A. Felfernig, P. Blazek, F.-C. Reinfrank, G. Ninaus, User interfaces for configuration environments, in A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen (Eds.), Knowledge-Based Configuration: From Research to Business Cases, Morgan Kaufmann, Elsevier, Waltham, MA, 2014, pp. 89-106.
- [3] L. L. Zhang, P. T. Helo, An empirical study on product configurators' application: implications, challenges, and opportunities, in J. Tiihonen, A. Falkner, T. Axling (Eds.), Proceedings of the 17th International Configuration Workshop, September 10-11, Vienna, Austria, 2015, pp. 5-10.
- [4] T. Leclercq, E. K. Abbasi, B. Dumas, M.-A. Remiche, P. Heymans, Essential expectations of users of web configurators: an empirical survey, *Proceedings of the ACM on Human-Computer Interaction* 6 (2022) 1-26.
doi: 10.1145/3534519.
- [5] J. Tidwell, Designing interfaces, 2nd ed., O'Reilly Media, Inc., Sebastopol, Canada, 2011.
- [6] I. Campo Gay, L. Hvam, Integrating user-centered practices in configuration systems development: framework and conceptual modelling, in M. Aldanondo, A. Falkner, A. Felfernig, M. Stettinger (Eds.), Proceedings of the 23rd International Configuration Workshop, Sep 16-17, Vienna, Austria, 2021, pp. 58-64.
- [7] ISO, ISO 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems, 2019. URL: <https://www.iso.org/standard/77520.html>.
- [8] F. F. Reichheld, T. Teal, The loyalty effect: the hidden force behind growth, profits and lasting, Boston, MA, 1996.
- [9] L. D. Hollebeek, M. S. Glynn, R. J. Brodie, Consumer brand engagement in social media: conceptualization, scale development and validation, *Journal of Interactive Marketing* 28 (2014) 149-165.
doi: 10.1016/j.intmar.2013.12.002.
- [10] A. Felfernig, M. Mandl, J. Tiihonen, M. Schubert, G. Leitner, Personalized user interfaces for product configuration, in C. Rich, Q. Yang, M. Cavazza, M. Zhou (Eds.), Proceedings of the 15th international conference on Intelligent user interfaces, Hong Kong, China, Association for Computing Machinery, New York, NY, 2010, pp. 317-320. doi: 10.1145/1719970.1720020.
- [11] J. A. Jacko (Ed.), Human computer interaction handbook: fundamentals, evolving technologies, and emerging applications, 3rd ed., CRC Press, Taylor & Francis Group, Boca Raton, FL, 2012.
- [12] M. Hassenzahl, N. Tractinsky, User experience - a research agenda, *Behaviour & Information Technology* 25 (2006) 91-97.
- [13] K. Pils, M. Enzelsberger, P. Ecker, Towards more flexible configuration systems: enabling product managers to implement configuration logic, in A. Felfernig, C. Forza, A. Haag (Eds.), Proceedings of the 16th International Configuration Workshop, September 25-26, Novi Sad, Serbia, 2014, pp. 55-58.
- [14] J. Tiihonen, T. Männistö, A. Felfernig, Sales configurator information systems design theory, in A. Felfernig, C. Forza, A. Haag (Eds.), Proceedings of the 16th International Configuration Workshop, September 25-26, Novi Sad, Serbia, 2014, pp. 67-74.
- [15] L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, M. Meyer, G. Petrone, R. Schäffer, W. Schütz, M. Zanker, Customizing the interaction with the user in on-line configuration systems, in M. Aldanondo (Ed.), Proceedings of the Configuration Workshop of 2002 European Conference of Artificial Intelligence, July 22-23, Lyon, France, HAL, 2002, pp. 119-124.
- [16] L. Ardissono, A. Felfernig, G. Friedrich, D. Jannach, R. Schäfer, M. Zanker, Intelligent interfaces for distributed web-based product and service configuration, in N. Zhong, Y. Yao, J. Liu, S. Ohsuga (Eds.), Web Intelligence: Research and Development. First Asia-Pacific Conference, WI 2001, Maebashi City, Japan, October 23-26, 2001, Proceedings, Springer, Berlin, Heidelberg, 2001, pp. 184-188. doi: 10.1007/3-540-45490-X_21.
- [17] L. Hvam, K. Ladeby, An approach for the development of visual configuration systems, *Computers & Industrial Engineering* 53 (2007) 401-419. doi: 10.1016/j.cie.2007.05.004.
- [18] Q. Boucher, G. Perrouin, J.-M. Davril, P. Heymans, Engineering configuration graphical user interfaces from variability models, in J.-S. Sottet, A. García Frey, J. Vanderdonckt (Eds.), Human Centered Software Product Lines, Springer, Cham, Switzerland, 2017, pp. 1-46. doi: 10.1007/978-3-319-60947-8_1.
- [19] B. Schneiderman, C. Plaisant, M. S. Cohen, S. Jacobs, N. Elmquist, N. Diakopoulos, Designing the user interface: strategies for effective human-computer interaction, 6th ed., Pearson, Hoboken, NJ, 2016.
- [20] M. Van Welie, G. C. Van Der Veer, A. Eliëns, Breaking down usability, *Interact* (1999) 613-620.
- [21] L. L. Zhang, S. Shafiee, Developing separate or integrated configurators? A longitudinal case study, *International Journal of Production Economics* 249 (2022) 1-17.
doi: 10.1016/j.ijpe.2022.108517.
- [22] C. Forza, F. Salvador, Managing for variety in the order acquisition and fulfilment process: the contribution of product configuration systems, *International Journal of Production Economics* 76 (2002) 87-98. doi: 10.1016/S0925-5273(01)00157-8.
- [23] E. Sandrin, Synergic effects of sales-configurator capabilities on consumer-perceived benefits of mass-customized products, *International Journal of Industrial Engineering and Management* 8 (2017) 177-188.
- [24] E. Sandrin, A. Trentin, C. Grossi, C. Forza, Enhancing the consumer-perceived benefits of a mass-customized product through its online

doi: 10.1080/01449290500330331.

- sales configurator: an empirical examination, *Industrial Management & Data Systems* 117 (2017) 1295-1315. doi: 10.1108/IMDS-05-2016-0185.
- [25] A. Trentin, E. Perin, C. Forza, Sales configurator capabilities to avoid the product variety paradox: construct development and validation, *Computers in Industry* 64 (2013) 436-447. doi: 10.1016/j.compind.2013.02.006.
- [26] A. Trentin, E. Perin, C. Forza, Increasing the consumer-perceived benefits of a mass-customization experience through sales-configurator capabilities, *Computers in Industry* 65 (2014) 693-705. doi: 10.1016/j.compind.2014.02.004.
- [27] E. Sandrin, C. Forza, Z. Aničić, N. Suzic, C. Grossi, T. Aichner, A. Trentin, Shoe configurators: a comparative analysis of capabilities and benefits, in V. Modrak (Ed.), *Mass customized manufacturing: theoretical concepts and practical approaches*, CRC Press, Taylor & Francis Group, Boca Raton, FL, 2017, pp. 193-216. doi: 10.1201/9781315398983-10.
- [28] C. Grossi, C. Forza, Users' social-interaction needs while shopping via online sales configurators, *International Journal of Industrial Engineering and Management* 10 (2019) 139-154.
- [29] E. K. Abbasi, A. Hubaux, M. Acher, Q. Boucher, P. Heymans, The anatomy of a sales configurator: an empirical study of 111 cases, in C. Salinesi, M. C. Norrie, Ó. Pastor (Eds.), CAiSE'13 - 25th International Conference on Advanced Information Systems Engineering, June 17-21, Valencia, Spain, Springer-Verlag, Berlin, Heidelberg, Germany, 2013, pp. 162-177. doi: 10.1007/978-3-642-38709-8_11.
- [30] P. Blazek, C. Streichsbier, M. Partl, L. Skjelstad, User interface modifications in established product configurators, in S. Hankammer, K. Nielsen, F. T. Piller, G. Schuh, N. Wang (Eds.), *Customization 4.0: Proceedings of the 9th World Mass Customization & Personalization Conference (MCPC 2017)*, Aachen, Germany, November 20-21, 2017, Springer International Publishing, Cham, Switzerland, 2018, pp. 451-466.
- [31] P. Blazek, M. Partl, L. Skjelstad, Learnings from monitoring web-based product configurator approaches in the world of customizable products, in Z. Aničić, C. Forza (Eds.), *Proceedings of the 8th International Conference on Mass Customization and Personalization - Community of Europe (MCP-CE 2018)*, September 19-21, Novi Sad, Serbia, Faculty of Technical Sciences, 2018, pp. 21-26.
- [32] C. Streichsbier, P. Blazek, F. Faltin, W. Frühwirt, Are de-facto standards a useful guide for designing human-computer interaction processes? The case of user interface design for web based B2C product configurators, in R. H. Sprague Jr. (Ed.), *Proceedings of the 42nd Hawaii International Conference on System Sciences*, January 5-8, Waikoloa, Big Island, HI, 5-8 Jan. 2009, IEEE Computer Society, Los Alamitos, CA, 2009, pp. 1-7. doi: 10.1109/HICSS.2009.80.
- [33] C. Streichsbier, P. Blazek, M. Partl, The impact of the product configurator user interface on customer purchase decisions, in Z. Aničić, C. Forza (Eds.), *Proceedings of the Proceedings of the 6th International Conference on Mass Customization and Personalization in Central Europe (MCP-CE 2014)*, September 24-26, Novi Sad, Serbia, Faculty of Technical Sciences, 2014, pp. 190-194.
- [34] T. Aichner, P. Coletti, Customers' online shopping preferences in mass customization, *Journal of Direct, Data and Digital Marketing Practice* 15 (2013) 20-35. doi: 10.1057/ddmp.2013.34.
- [35] T. Aichner, B. Gruber, Managing customer touchpoints and customer satisfaction in B2B mass customization: a case study, *International Journal of Industrial Engineering and Management* 8 (2017) 131-140.
- [36] T. Aichner, A. M. Shaltoni, The impact of perceived advertising creativity on behavioural intentions and quality perceptions in mass customization, *International Journal of Industrial Engineering and Management* 10 (2019) 131-138. doi: 10.24867/IJIEM-2019-2-234.
- [37] European Union, Key competences for lifelong learning: a European reference framework, *Official Journal of the European Union* 61 (2018) 7-13. doi: 10.2766/569540.
- [38] M. Noack, Unpacking transversal skills and competences, Bertelsmann Stiftung, 2021.
- [39] European Commission, ESCO: European Skills, Competences, Qualifications and Occupations, Accessed 2023. URL: esco.ec.europa.eu/en.
- [40] ISTAT, Nomenclatura e classificazione delle unità professionali, Accessed 2023. URL: professioni.istat.it/sistemainformativoprofessioni/cp2011/.
- [41] Istituto Nazionale per l'Analisi delle Politiche Pubbliche, Atlante del lavoro, Accessed 2022. URL: atlantelavoro.inapp.org/atlante_lavoro.php.
- [42] C. Forza, F. Salvador, HRM policies for mass customization: understanding individual competence requirements and training needs for the mass customizing industrial company, in T. Blecker, G. Friedrich (Eds.), *Mass Customization: Challenges and Solutions*, Springer, New York, NY, 2006, pp. 251-269. doi: 10.1007/0-387-32224-8_12.
- [43] A. Trentin, T. Somià, E. Sandrin, C. Forza, Operations managers' individual competencies for mass customization, *International Journal of Operations & Production Management* 39 (2019) 1025-1052. doi: 10.1108/IJOPM-10-2018-0592.
- [44] E. Sandrin, C. Forza, G. Leitner, A. Trentin, Configuration manager: describing an emerging professional figure, in A. Felfernig, L. Fuentes (Eds.), *Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume B*, Graz, Austria, Association for Computing Machinery, New York, NY, 2022, pp. 193-200. doi: 10.1145/3503229.3547049.

Appendix

Table 1

Placement of the “user interface expert for configurators” among the ESCO profiles, the NUP ISTAT profiles, and areas of activities of the “Atlante del Lavoro”

ESCO Profile ^a (in English, German, Italian)	NUP ISTAT (ISTAT CP2011) profiles ^b	“Aree di Attività” (ADA) from “Atlante del Lavoro” ^c (Relevant activities for the UI expert for configurators)
2513.3 - User interface designer - Multimedia-Designer/Multimedia-Designerin - Progettista di interfacce utente	2.7.1.1.1 - Analisti e progettisti di software 2.7.1.1.2 - Analisti di sistema	ADA.14.01.01 (ex ADA.16.237.773) - Definizione e implementazione della strategia organizzativa nell'ICT ADA.14.01.02 (ex ADA.16.237.775) - Identificazione e definizione delle proposte per lo sviluppo dei servizi IT ADA.14.01.03 (ex ADA.16.238.776) - Gestione del processo di sviluppo del business in ambito Information Technology ADA.14.01.04 (ex ADA.16.238.777) - Allineamento tra strategie di business e sviluppo tecnologico ADA.14.01.05 (ex ADA.16.238.778) - Ideazione e definizione della specifica soluzione ICT ADA.14.01.06 (ex ADA.16.238.779) - Supporto al cliente per l'innovazione nell'ICT ADA.14.01.12 (ex ADA.16.239.785) - Progettazione e realizzazione di applicativi software multi-tier ADA.14.01.14 (ex ADA.16.239.787) - Progettazione e realizzazione dell'interfaccia utente ADA.14.01.16 (ex ADA.16.239.789) - Deployment, integrazione e verifica della soluzione ICT
2512.5 - User interface developer - Entwickler von Benutzeroberflächen/Entwicklerin von Benutzeroberflächen - Sviluppatore di interfacce utente	2.7.1.1.3 - Analisti e progettisti di applicazioni web	
2511.19 - User experience analyst - User Experience Analyst - Analista della user experience		
3512 - Information and communications technology user support technicians - Techniker für die Anwenderbetreuung in der Informations- und Kommunikationstechnologie - Tecnici per l'assistenza agli utenti della tecnologia dell'informazione e della comunicazione	3.1.2.2.0 - Tecnici esperti in applicazioni 3.1.2.3.0 - Tecnici web	ADA.14.01.06 (ex ADA.16.238.779) - Supporto al cliente per l'innovazione nell'ICT ADA.14.01.12 (ex ADA.16.239.785) - Progettazione e realizzazione di applicativi software multi-tier ADA.14.01.14 (ex ADA.16.239.787) - Progettazione e realizzazione dell'interfaccia utente ADA.14.01.16 (ex ADA.16.239.789) - Deployment, integrazione e verifica della soluzione ICT
3514.1 - Webmaster - Webmaster/Webmasterin - Webmaster		
2431.15 - Product and services manager - Produkt- und Dienstleistungsmanager/Produkt- und Dienstleistungsmanagerin - Responsabile beni e servizi	2.5.1.5.2 - Specialisti nella commercializzazione di beni e servizi (escluso il settore ICT) 2.5.1.5.3 - Specialisti nella commercializzazione nel settore delle tecnologie dell'informazione e della comunicazione	ADA.14.01.01 (ex ADA.16.237.773) - Definizione e implementazione della strategia organizzativa nell'ICT

^a URL: <https://esco.ec.europa.eu/en>

^b URL: <https://professioni.istat.it/sistemaninformativoprofessioni/cp2011/>

^c URL: https://atlantelavoro.inapp.org/atlante_lavoro.php

Specifying Configurable Videos with Feature Models

Sebastian Lubos¹, Alexander Felfernig¹ and Viet-Man Le¹

¹Graz University of Technology, Graz, Austria

Abstract

Personalization of products is a popular aspect in various application domains, including videos. Enabling users to consume personalized learning videos that include only individually relevant content has the potential to deliver additional benefits in e-learning, for example, by making learning more efficient. In this paper, we present a practical approach to define configurable videos based on feature models, as well as an integrated solution to derive personalized videos by respecting given constraints. Additionally, the possibility to extend the configuration with interactive video elements is described to enable an improved user experience.

Keywords

Feature Models, Configuration, Interactive Video, Personalized Video, Video Summarizing,

1. Introduction

Configuration of software, services, and products fulfilling individual needs has been a popular topic of research in recent years [1, 2]. Feature models [3] have thereby proven to be an excellent approach to solving those challenges in a variety of domains [4], including videos [5]. The possibility to create personalized videos offers huge potential, especially in the domains of knowledge transfer and e-learning where videos are consumed to increase know-how or study new topics.

A challenge in this area is the availability of different learning videos on video platforms, for example, YouTube¹, covering various topics, explained to consumers with individual pre-knowledge. The large variety of options and poor possibilities to determine if videos are relevant before consumption make it complicated for users to find adequate videos [6]. Effective learning videos have the characteristics to include all relevant information for a user to understand and follow the video. At the same time, they reduce learning time by excluding unrelated or already known information [5].

The usage of natural language queries to retrieve video summaries [7], and more recently the integration of chatbots to query and interact with videos², have been published as a possibility to support users. While those approaches work well if users are able to specify what they are searching for, a knowledge-based configuration of videos [5] has been presented as a possibility to mitigate

this weakness, by giving the user more assistance. User requirements are collected and used in a *Constraint Satisfaction Problem* (*CSP*) to determine a video fulfilling the user requirements.

Based on the findings in [5], we demonstrate a flexible and reusable approach to define configurable videos, and show an example instantiation that provides a personalized video using the *Choco* solver³.

Previous work in the synthetic creation of videos has applied video processing techniques to change the visual content to generate artwork variants using variability management techniques [8], and to generate multiple video variants for algorithm test samples [9, 10, 11]. In our approach, we preserve the video content of existing videos and parts of videos while enhancing user experience by transforming it into a well-organized structure. This extends the work of an online video generator taking an initial selection of video clips as seed to create variants of humorous video segments [12], by applying it in the domain of learning videos with additional user requirements and more complex constraints.

The major contributions of the paper are the following. We extend our previous work on the problem definition of configurable videos [5], by demonstrating a practical implementation. A reusable and adaptable approach to defining the structure of configurable videos using feature model technologies is presented, including the integration of a solver to generate personalized videos with respect to specified user requirements. Furthermore, we explain how the solution can be extended to integrate decision points with interactive video elements [13] for an improved individual user experience.

The remainder of this paper is organized as follows. Our approach to specifying a configurable video is explained in Section 2. In Section 3, we present an example configuration with specified user requirements and the resulting video. In Section 4, the usability and limita-

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

✉ slubos@ist.tugraz.at (S. Lubos); alexander.felfernig@ist.tugraz.at (A. Felfernig); vietman.le@ist.tugraz.at (V. Le)

>ID 0000-0002-5024-3786 (S. Lubos); 0000-0003-0108-3146 (A. Felfernig); 0000-0001-5778-975X (V. Le)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://www.youtube.com>

²e.g., <https://www.ortusbuddy.ai/>

³<https://choco-solver.org>

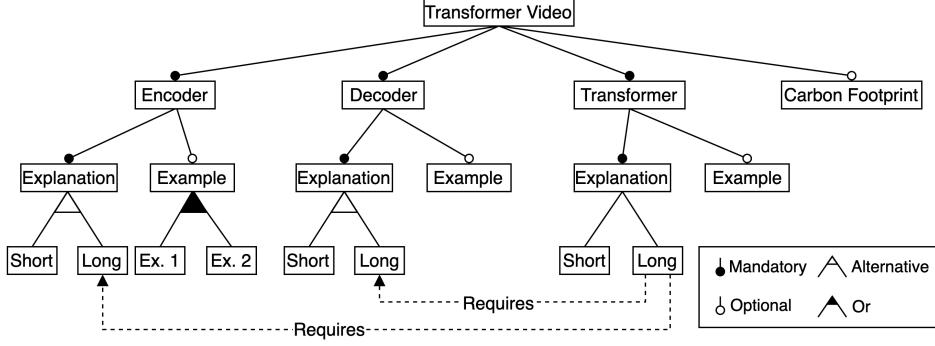


Figure 1: Feature model of an example configurable video about the transformer model.

tion of this approach are discussed. Finally, open research issues are discussed in Section 5 before the paper is concluded with Section 6.

2. Specification of Configurable Videos

Developing feature models is a complex task for persons not experienced with the notation and technologies. In order to ease the problem definition, we enable users to define the structure of configurable videos using the JSON notation provided in [14]. The JSON format is heavily used in different applications, and many software developers have experience with it. For this reason, we expect that it will ease the future implementation of a GUI-based editor for configurable videos, such that they can be configured by everyday users.

Within this paper, we use the configuration of a learning video explaining the *transformer* model in machine learning [15] as a running example. A transformer model is a type of deep learning architecture designed to process sequential data, such as text or speech, by leveraging self-attention mechanisms. It is a popular and powerful model for a variety of natural language processing (NLP) applications, including, machine translation, language understanding, and text generation.

For our example, we use videos from the *Hugging Face* tutorial on NLP⁴ hosted on YouTube. Using those tutorial videos, we designed a configurable video with the structure presented as feature model [3] in Figure 1. Leaves of the model refer to single video segments and parent features define the category. Video segments are interpreted as the property values of the configured video. Following the structure, we see that each video includes at least an explanation of the individual components of the transformer model (*encoder* and *decoder*), followed

by an explanation of the overall model. Examples of each part can be included optionally, as well as a digression on the carbon footprint of transformer models.

Using the formatting described in [14] the feature model can be described in a JSON format using a straightforward approach. The nested JSON structure enables the user to define the model following a top-down approach, where for each node, an id, the type (mandatory, optional, root), child nodes as well as sibling-relation (alternative, or) can be defined. Furthermore, exclusions and required properties can be specified using their id. Following the algorithms described in [14], the JSON structure can be translated into a valid feature model. In Figure 2, a part of the configuration is shown as an example.

In addition to this generic approach to describing the structure of the configurable video, information on the included video segments is required. More specifically, an *URL* where the video is available is needed. Additionally, the duration of the segments in seconds is required to define constraints for the overall duration of the generated video. This is also defined using a JSON structure, where each key references an id of the model described as JSON. An example is shown in Figure 3.

The definition of the configured video is then used to instantiate a model for the CSP. We used *PyCSP*⁵ for this purpose, which is a *Python* framework, to describe models for *CSPs* in a declarative manner. It includes the possibility to choose between the solver of *ACE* (*AbsCon Essence*)⁶ and *Choco*. The generic code to use provided JSON files for the instantiation of the configurable video, as well as the complete example, are available in our repository⁷.

Besides the description of a configurable video, user requirements need to be collected in order to personalize the video. While different possibilities, including the assessment of pre-knowledge, are possible, we restrict

⁵<http://pycsp.org>

⁶<https://github.com/xcsp3team/ace>

⁷<https://github.com/slubos/specifying-configurable-videos>

⁴<https://huggingface.co/learn/nlp-course>

```
{
  "id": "TransformerVideo",
  "type": "root",
  "parent": "",
  "relation": "",
  "requires": [],
  "excludes": [],
  "children": [
    {
      "id": "Encoder",
      "type": "mandatory",
      "parent": "TransformerVideo",
      "relation": "",
      "requires": [],
      "excludes": [],
      "children": [
        {
          "id": "EncoderExplanation",
          "type": "mandatory",
          "parent": "Encoder",
          "relation": "",
          "requires": [],
          "excludes": [],
          "children": [
            {
              "id": "EncoderExplanationShort",
              "type": "optional",
              "parent": "EncoderExplanation",
              "relation": "alternative",
              "requires": [],
              "excludes": [],
              "children": []
            },
            ...
          ],
          ...
        },
        ...
      ],
      ...
    },
    ...
  ],
  ...
}
```

Figure 2: A part of the JSON definition of the configuration aspects for the example video on transformer models.

those to the maximum video duration for our example, assuming that the video is suitable for a beginner level. Especially in preparation for exams, students often follow the utility maximization problem [16], and try to learn as much as possible in a limited amount of time. To capture this requirement, the maximum acceptable video duration of a user is collected and translated to a maximum duration constraint. The duration is determined by summing the duration of each included video segment.

The personalized video is then generated following the *configuration task* described in [5], consisting of a feature

```
{
  "EncoderExplanationShort": {
    "url": "https://youtu.be/H39Z...",
    "duration": 45
  },
  "EncoderExplanationLong": {
    "url": "https://youtu.be/MUqN...",
    "duration": 141
  },
  ...
}
```

Figure 3: A part of the JSON definition of video sources for the example video on transformer models.

model and a defined set of user requirements [17]. The solution of this task is a *configuration*, i.e., an assignment of variables in the CSP, such that the constraints of the model and user requirements are fulfilled [5].

In our presented approach, the variables V state which of the individual video segments are included in the configured video. The respective variable domain is $\{true, false\}$, describing the inclusion or exclusion of a video segment. Nodes of the feature model described in the JSON file are defined as variables. Furthermore, for all video segments, a variable describing the duration is defined within the domain $\{0, \text{videoduration}\}$, where *videoduration* is the length of the video in seconds defined in the JSON file. Using a constraint, the value of this variable is restricted to 0 or the defined *videoduration* depending on the inclusion of the segment in the configured video. Further knowledge base constraints are directly derived from the feature model described as JSON file, following the algorithms described in [14].

3. Video Configuration Results

A configured video can be interpreted as a simple playlist, i.e., the included video segments will be played in an ordered fashion. Table 1 shows an example of the minimal and maximal video configuration in terms of video duration of the transformer model example. Depending on the maximum acceptable video duration of the user, different video segments are included or excluded.

Since multiple versions can be the result of a configuration task, the user has the choice to select one of the options. Considering, for example, 250s as the maximum acceptable duration, 15 configurations have been found. To enable the choice, the total duration could be shown, such that the user can select if they want to use most of their available time or not. Alternatively, a further explanation alternative might describe the content in a contrastive way, for example, *video A* contains more detailed explanations, while *video B* has more examples.

Transformer Video												
	Encoder				Decoder			Transformer			Carbon Foot-print	
	Explanation		Example		Explanation		Example	Explanation		Example		
	Short	Long	Ex. 1	Ex. 2	Short	Long		Short	Long			
Min. Config	X				X			X				
Max. Config		X	X	X		X	X		X	X	X	

Table 1

The examples demonstrate the minimal and maximal video configurations in terms of video duration for the transformers model example. An X indicates that the respective video segment is part of the configuration.

A more advanced solution is based on *interactive videos* which offer an extension to classical videos by offering several interactivity features [13]. This approach is used to enable the user during the video consumption if parts of the video should be included, as long as they still fulfill the duration requirement. In terms of alternative videos, this means a choice is presented which path is followed. After each choice, the selection is included as a constraint, and new paths are configured on the fly. For "or" video segments, the user can have the option to choose one or both. In the case of optional segments, the user is asked if they want to watch it.

Figure 4 sketches the possible path flow including the decision points for the transformer video example described with the feature model in Figure 1, given the example requirement of 250s as the maximum acceptable duration. Decision points in the workflow diagram are shown with the diamond symbol. For an interactive video, this can be implemented as a question, with choices shown by the outgoing arrows, labeled with their description. The rectangle with rounded corners indicates which video is played. A circle indicates the start, while a double-edged circle represents the end.

Each time a user takes a decision, the value is added as a constraint to the CSP, such that the remaining paths and options are computed dynamically while the user is consuming the video.

4. Discussion

This paper presents a reusable approach for creating configurable videos adaptable to any topic. It requires the availability of manually structured videos by the creator, and the specification of video sources must be updated accordingly (see Figure 3). While YouTube videos were used as an example, any video source could be utilized. The video creator is responsible for adapting the model for the configurable video to represent their desired structure and constraints (see Figure 2). This approach is versatile and can represent any video structure, also including more videos and constraints. We expect the approach to scale well for more complex videos, as the constraints are

rather simple and the number of variables is manageable. Yet, we leave this experiment open for future work.

To simplify the process of specifying configurable videos, a GUI-based editor could be used instead of relying on the video creator's knowledge of JSON files and feature models. The editor would allow the user to add videos by pasting links, then organize them into categories using a drag-and-drop approach. Constraints within categories could be specified using different group types, indicating whether they are alternatives or multiple options to include. Additionally, the video creator could designate videos as mandatory or optional. Cross-tree constraints could be added additionally to specify the requirement of videos from other categories. We expect this user-friendly approach to be easily understandable, eliminating the need for understanding feature models. The translation of the GUI input to JSON is handled by the application.

5. Open Issues for Future Work

One topic for future work is the implementation of the interactive video approach described in this paper. Frameworks for this purpose, e.g., FrameTrail⁸ or H5P⁹, offer the possibility to define the interactive elements and use them for playout. As we expect that this kind of video consumption improves learning effectiveness, conducting a user study to examine this assumption is planned. A between-subject study could be conducted, where one group uses interactive videos, while the other views the complete video without interaction. Questionnaires about the video topic immediately after the video and after a few weeks could be used to analyze the short- and long-term learning effectiveness of the interactive approach.

Further topics include the support of users in the definition of configurable videos. While our approach offers the possibility to describe the structure of those, it is still a lot of manual annotation work to describe the video,

⁸<https://frametrail.org>

⁹<https://h5p.org/>

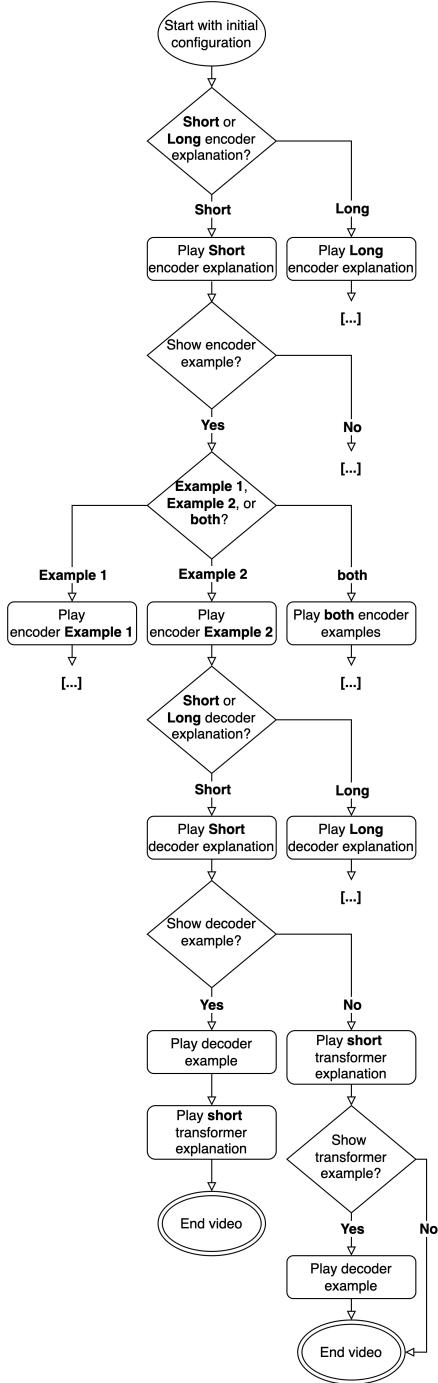


Figure 4: Possible paths of the configured video example with interactive elements. Diamonds indicate decisions users take during video consumption, while rectangles resemble played video segments. Not all paths are shown for simplicity.

for this purpose, options to automatize parts of this work should be considered. This includes the automated indexing of video content to ease recognition of what is included [18, 19], as well as their semantic segmentation defining the individual video segments that can be included ([20, 21]). Also, the possible inclusion of recommendation technologies [22, 23] to support the definition of those videos, e.g., by recommending which options could further be included.

Finally, the inclusion of diagnoses [24, 25] should be considered to relax situations where no solution can be found for given user requirements. Those can help to find a configuration that takes into account as much as possible of the original user requirements.

6. Conclusions

With this paper, we present an initial implementation to define and generate personalized videos using feature models. Using an easy-to-use JSON notation, a solution was presented that is able to add additional benefit to knowledge transfer with videos by reusing already existing material. Following a practical example learning video, we showed how the approach can be used, and further extended to enable its usage with interactive videos, which is part of our future work.

Acknowledgments

The presented work has been developed within the research project Streamdiver which is funded by the Austrian Research Promotion Agency (FFG) under the project number 886205.

References

- [1] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration - From Research to Business Cases, Elsevier, 2014.
- [2] D. Sabin, R. Weigel, Product configuration frameworks - a survey, IEEE Intelligent Systems 13 (1998) 42–49.
- [3] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-oriented Domain Analysis (FODA) – Feasibility Study, Technical Report CMU – SEI-90-TR-21 (1990).
- [4] J. Martinez, W. K. G. Assunção, T. Ziadi, Espla: A catalog of extractive spf adoption case studies, in: Proceedings of the 21st International Systems and Software Product Line Conference - Volume B, SPLC ’17, Association for Computing Machinery, New York, NY, USA, 2017, p. 38–41. URL: <https://doi.org/10.1145/3052973.3053000>.

- org/10.1145/3109729.3109748. doi:10.1145/3109729.3109748.
- [5] S. Lubos, M. Tautschnig, A. Felfernig, V.-M. Le, Knowledge-based configuration of videos using feature models, in: Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume B, SPLC '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 188–192. URL: <https://doi.org/10.1145/3503229.3547052>. doi:10.1145/3503229.3547052.
- [6] A. Imran, F. Alaya Cheikh, S. Kowalski, Automatic annotation of lecture videos for multimedia driven pedagogical platforms, *Knowledge Management and E-Learning* 7 (2015).
- [7] M. Vahedi, M. M. Rahman, F. Khomh, G. Uddin, G. Antoniol, Summarizing relevant parts from technical videos, in: 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 2021, pp. 434–445. doi:10.1109/SANER50967.2021.00047.
- [8] J. Martinez, G. Rossi, T. Ziadi, T. F. D. A. Bissyandé, J. Klein, Y. Le Traon, Estimating and predicting average likability on computer-generated artwork variants, in: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 1431–1432. URL: <https://doi.org/10.1145/2739482.2764681>. doi:10.1145/2739482.2764681.
- [9] M. Acher, M. Alférez, J. A. Galindo, P. Romenteau, B. Baudry, Vivid: A variability-based tool for synthesizing video sequences, in: Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools - Volume 2, SPLC '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 143–147. URL: <https://doi.org/10.1145/2647908.2655981>. doi:10.1145/2647908.2655981.
- [10] J. A. Galindo, M. Alférez, M. Acher, B. Baudry, D. Benavides, A variability-based testing approach for synthesizing video sequences, in: Proceedings of the 2014 International Symposium on Software Testing and Analysis, ISSTA 2014, Association for Computing Machinery, New York, NY, USA, 2014, p. 293–303. URL: <https://doi.org/10.1145/2610384.2610411>. doi:10.1145/2610384.2610411.
- [11] M. Alférez, M. Acher, J. A. Galindo, B. Baudry, D. Benavides, Modeling variability in the video domain: Language and experience report, *Software Quality Journal* 27 (2019) 307–347. URL: <https://doi.org/10.1007/s11219-017-9400-8>. doi:10.1007/s11219-017-9400-8.
- [12] G. Bécan, M. Acher, J.-M. Jézéquel, T. Menguy, On the variability secrets of an online video generator, in: Proceedings of the Ninth International Workshop on Variability Modelling of Software-Intensive Systems, VaMoS '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 96–102. URL: <https://doi.org/10.1145/2701319.2701328>. doi:10.1145/2701319.2701328.
- [13] A. Palaigeorgiou, George and Papadopoulou, I. Kazanidis, Interactive video for learning: A review of interaction types, commercial platforms, and design guidelines, in: M. Tsitouridou, J. A. Diniz, T. A. Mikropoulos (Eds.), *Technology and Innovation in Learning, Teaching and Education*, Springer International Publishing, Cham, 2019, pp. 503–518.
- [14] H. Shatnawi, H. C. Cunningham, Encoding feature models using mainstream json technologies, in: Proceedings of the 2021 ACM Southeast Conference, ACM SE '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 146–153. URL: <https://doi.org/10.1145/3409334.3452048>. doi:10.1145/3409334.3452048.
- [15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, *Advances in neural information processing systems* 30 (2017).
- [16] A. Mas-Colell, M. D. Whinston, J. R. Green, et al., *The Utility Maximization Problem*, volume 1, Oxford university press New York, 1995.
- [17] L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley, K. Wolter, *Configuration Knowledge Representation and Reasoning*, 1 ed., Elsevier B.V., Netherlands, 2014, pp. 41–72.
- [18] Y. Deldjoo, *Enhancing Video Recommendation Using Multimedia Content*, Springer International Publishing, Cham, 2020, pp. 77–89. URL: https://doi.org/10.1007/978-3-030-32094-2_6. doi:10.1007/978-3-030-32094-2_6.
- [19] M. Elahi, F. Bakhshandegan Moghaddam, R. Hosseini, M. H. Rimaz, N. El Ioini, M. Tkalcic, C. Trattner, T. Tillo, *Recommendng Videos in Cold Start With Automatic Visual Tags*, Association for Computing Machinery, New York, NY, USA, 2021, p. 54–60. URL: <https://doi.org/10.1145/3450614.3461687>.
- [20] T. Tuna, M. Joshi, V. Varghese, R. Deshpande, J. Subhlok, R. Verma, Topic based segmentation of classroom videos, in: 2015 IEEE Frontiers in Education Conference (FIE), 2015, pp. 1–9. doi:10.1109/FIE.2015.7344336.
- [21] P. A. Co, W. R. Dacuyan, J. G. Kandt, S.-C. Cheng, C. L. Sta. Romana, Automatic topic-based lecture video segmentation, in: *Innovative Technologies and Learning: 5th International Conference, ICITL 2022, Virtual Event, August 29–31, 2022, Proceedings*, Springer-Verlag,

- Berlin, Heidelberg, 2022, p. 33–42. URL: https://doi.org/10.1007/978-3-031-15273-3_4. doi:10.1007/978-3-031-15273-3_4.
- [22] A. Falkner, A. Felfernig, A. Haag, Recommendation Technologies for Configurable Products, *AI Magazine* 32 (2011) 99–108.
 - [23] A. Felfernig, V.-M. Le, A. Popescu, M. Uta, T. N. T. Tran, M. Atas, An overview of recommender systems and machine learning in feature modeling and configuration, in: 15th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS’21, Association for Computing Machinery, New York, NY, USA, 2021. URL: <https://doi.org/10.1145/3442391.3442408>. doi:10.1145/3442391.3442408.
 - [24] A. Felfernig, M. Schubert, C. Zehentner, An efficient diagnosis algorithm for inconsistent constraint sets, *AI for Engineering Design, Analysis, and Manufacturing (AIEDAM)* 26 (2012) 53–62.
 - [25] A. Felfernig, R. Walter, J. Galindo, D. Benavides, M. Atas, S. Polat-Erdeniz, S. Reiterer, Anytime Diagnosis for Reconfiguration, *Journal of Intelligent Information Systems* 51 (2018) 161–182.

Solving Constraint Satisfaction Problems with Database Queries: An Overview

Alexander Felfernig¹, Viet-Man Le¹, Albert Haag² and Sebastian Lubos¹

¹Institute of Software Technology, Graz University of Technology, Graz, Austria

²Product Management Haag GmbH, Bad Dürkheim, Germany

Abstract

Knowledge-based configuration tasks are often solved on the basis of constraint programming. Using constraint programming requires technical expertise regarding problem specification and – to some extent – also solution search, for example, in terms of being confronted with the definition of search heuristics. In this paper, we show how to apply database queries to solve knowledge-based configuration tasks. Using this approach, configuration tasks can be defined and solved without the need of integrating a potentially new technology, but rather stick with technical infrastructures (i.e., relational databases) already existing in the company.

Keywords

Constraint Solving, Knowledge-based Configuration, Database Queries

1. Introduction

Constraint programming (CP) [1] is based on the idea of defining a set of problem variables, variable domains, and related restrictions (constraints) and solving the problem using a constraint solver. As such, this technology is often used for solving configuration tasks [2, 3]. There are further approaches used for configuration knowledge representation. For example, SAT solving [4] is based on the idea of representing a configuration task by set of Boolean variables where each variable represents a variable domain value in the general constraint satisfaction problem, for example, car color *red* is a domain value of the variable *color*. In the SAT context, *red* would be regarded as variable with the domain {*true*, *false*}. In addition, answer set programming (ASP) is based on a more object-oriented view on configuration knowledge representation [5] where on the reasoning level, ASP programs are solved using SAT solvers.

All these types of knowledge representation require additional expertise in at least one of the areas of constraint programming or SAT solving. Furthermore, additional investments are needed to increase CP-related knowledge of employees which is a major precondition for making underlying technologies applicable for configuration knowledge representation and reasoning. On the other hand, relational database technologies and related

query languages are wide-spread in industrial software development projects. Our idea is to exploit the same technologies in a different form for the representation and solving of constraint-based configuration tasks. In this paper, we provide an overview of different types of knowledge representations and corresponding database queries that can be used to support configuration tasks. We focus on specific database queries which can be regarded as a conjunction of the individual constraints of a corresponding constraint satisfaction problem.

The contributions of this paper are the following: (1) we introduce the idea of a configuration task and a corresponding configuration defined and determined on the basis of the concepts of database queries. (2) we discuss the results of performance evaluations with existing configuration benchmark bases.

The remainder of this paper is organized as follows. In Section 2, we introduce an example of a car configuration task defined in terms of a constraint satisfaction problem (CSP). In Section 3, we introduce a database query based definition of a configuration task and discuss possibilities of table-based configuration knowledge representations. Thereafter, in Section 4 we provide a performance comparison between database query based and constraint solving based configuration. Threats to validity are discussed in Section 5. Finally, the paper is concluded with a summary of open research issues in Section 6.

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

✉ alexander.felfernig@ist.tugraz.at (A. Felfernig); vietman.le@ist.tugraz.at (V. Le); albert@product-management-haag.de (A. Haag); slubos@ist.tugraz.at (S. Lubos)
↳ 0000-0003-0108-3146 (A. Felfernig); 0000-0001-5778-975X (V. Le); 0000-0002-1388-4904 (A. Haag); 0000-0002-5024-3786 (S. Lubos)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

2. Example Configuration Task

Following the concepts of constraint programming [1], a configuration task can be defined in terms of (1) finite domain variables $v_i \in V = \{v_1..v_n\}$ (including the corresponding domain definitions $dom(v_i)$) describing product properties and user preferences and (2) constraints

Table 1

Implicit configuration space description where each (CSP) variable $v_i \in V$ is represented by a single table with one attribute val and the table entries derived from $dom(v_i)$, for example, variable $type$ is represented by the corresponding table $type$. The corresponding database query has to take into account all constraints in C .

table	type	fuel	skibag	4wheel	pdc
attribute	val	val	val	val	val
domain	city,limo,combi,xdrive	4l,6l,10l	yes,no	yes,no	yes,no

$C = CF \cup CR$ representing product domain-specific constraints CF and customer requirements CR [2].

A simplified example of a car configuration task is the following where $type$ represents the car type, $fuel$ represents the average fuel consumption, $skibag$ indicates the availability of a skibag, and pdc represents a parc distance control feature. In this example, the product domain specific constraints are $CF = \{c_1..c_5\}$ and the customer requirements are $CR = \{c_6..c_9\}$ which can be specified in a complete (all variables in V have a value) or incomplete fashion.

- $V = \{type, fuel, skibag, 4wheel, pdc\}$
- $dom(type) = \{city, limo, combi, xdrive\}. dom(fuel) = \{4l, 6l, 10l\}. dom(skibag) = \{yes, no\}. dom(4wheel) = \{yes, no\}. dom(pdc) = \{yes, no\}.$
- $CF = \{c_1 : 4wheel = yes \rightarrow type = xdrive, c_2 : skibag = yes \rightarrow type \neq city, c_3 : fuel = 4l \rightarrow type = city, c_4 : fuel = 6l \rightarrow type \neq xdrive, c_5 : type = city \rightarrow fuel \neq 10l\}$
- $CR = \{c_6 : 4wheel = yes, c_7 : fuel = 6l, c_8 : type = city, c_9 : skibag = yes\}$

Based on this example CP-based configuration task representation, we will now discuss in more detail different options to represent and solve a configuration task on the basis of a corresponding database query definition.

3. Database Query Based Configuration

Representing a configuration task on the basis of a database query allows for the application of relational database technologies for determining corresponding configurations (solutions). We now introduce a definition of a configuration task on the basis of a database query setting $P_{[C]}S$ (see Definition 1). In this context, P can be (1) a set of tables where each table represents one $v_i \in V$ (“one table per variable” representation), (2) one table including “all possible configurations” (solutions), and (3) a set of tables representing tuples consistent with individual constraints in $c_i \in C$ (“local consistency”). Furthermore, $[C]$ is the set of constraints representing the selection criteria of the database query. Finally, S includes those variables $v_i \in V$ representing the projection criteria

of the query, i.e., those variable values that should be shown as result of the configuration task.

Definiton 1 (Configuration Task). A *configuration task* can be defined as database query $P_{[C]}S$. In this context, P represents all possible configurations in tabular form (explicitly or implicitly) and $[C]$ represents the selection criteria of the query in conjunctive form, i.e., $\bigwedge_{(c_i \in C)} c_i$. Furthermore, S represents the projection attributes (variables). In this context, $C = CR \cup CF$ with CR representing the given customer requirements and CF representing product domain-specific constraints.

Given such a definition of a configuration task, we are now able to introduce the definition of a configuration (see Definition 2).

Definiton 2 (Configuration). A *configuration* for configuration task is one tuple of a result of executing a database query $P_{[C]}S$ using the selection criteria in C and the projection attributes (variables) in S .

Based on Definitions 1–2, we now discuss different ways of representing configuration knowledge in a tabular fashion. The chosen type of knowledge representation has an impact on the way the corresponding database query has to be formulated – for demonstration purposes, we will include related examples.

(1) “One Table per Variable”. Using this representation, configuration knowledge is expressed in terms of tables (representing individual CSP variables) – see Table 1. For example, the CSP variable $type$ is represented by table $type$ with the attribute val having the CSP variable domain expressed by individual tuples $\{(city), (limo), (combi), (xdrive)\}$. Following this type of representation, the database (SQL) query $P_{[C=CF \cup CR]}S$ for our car configuration task is the following (see Query 1).

```
1. SELECT * FROM type,fuel,skibag,4wheel,pdc
   WHERE (not 4-wheel.val=yes or type.val=xdrive)
         and .. and (skibag.val=yes).
```

In this example, we assume that S includes all variables in V and P is regarded as implicit representation of the Cartesian product $type \times fuel \times skibag \times 4wheel \times pdc$.¹ Table 2 shows one configuration returned by Query 1. The

¹We want to assure that at most one tuple is returned by a query – to support this, we assume a query setting such as $LIMIT=1$ (this is database-specific).

configuration includes the pdc feature, i.e., $pdc.val = yes$ (this attribute has not been specified by the user).

Table 2

Configuration determined by Query 1.

table	type	fuel	skibag	4wheel	pdc
val	xdrive	6l	yes	yes	yes

Note that if we are interested only in specific attribute values, the query projection has to specify those attributes, for example, since $4wheel$, $fuel$, $type$, and $skibag$ have already been specified as customer requirements, only pdc needs to be included (see Query 2).

2. SELECT DISTINCT $pdc.val$
FROM $type$, $fuel$, $skibag$, $4wheel$, pdc
WHERE (not $4-wheel.val=yes$ or $type.val=xdrive$)
and .. and ($skibag.val=yes$).

Example Query Optimization. Possibilities of improving the performance of such queries are (1) to reduce variable domains in terms of assuring node consistency (e.g., each value of the domain of the variable $type$ must be consistent with each unary constraint referring to this variable). (2) queries can be “enriched” by including so-called no-goods (conflict sets) [6] in negated form – the determination of possible conflicts must also be performed in a pre-calculation step. (3) It is also possible to further restrict variable (attribute) domains by establishing arc consistency within a pre-calculation step.

(2) “All Possible Configurations”. Specifically for small configuration problems with a limited configuration space size there is also the possibility of just enumerating all possible configurations and storing those configurations in a corresponding table (see, e.g., Table 3). Such an enumeration can be performed on the basis of a database query $P_{[CF]}S$ where P is a table that includes all possible configurations and CF represents the set of domain-specific constraints.

Table 3

Explicit configuration space description in one table P including the CSP variables $v_i \in V$ as table attributes. The corresponding database query has to take into account the constraints in CR (CF is already taken into account in P).

type	fuel	skibag	4wheel	pdc
xdrive	6l	yes	yes	yes
xdrive	6l	yes	yes	no
..
city	4l	no	no	no

Following this knowledge representation, the database query in the context of our car configuration task is the following (see Query 3). In this example, we again assume that S includes all CSP variables. Furthermore, P

represents a table that includes all (pre-generated) possible configurations.

3. SELECT * FROM P
WHERE (4-wheel=yes) and .. and (skibag=yes).

Table 4 shows one configuration returned by Query 3.

Table 4

Configuration determined by Query 3.

type	fuel	skibag	4wheel	pdc
xdrive	6l	yes	yes	yes

Example Query Optimization. A basic approach to increase query efficiency is to reduce the number of table entries in P . For example, instead of having one centralized table, we could introduce one table per car $type$ which is reasonable if the user is sure about the car type selection and just wants to configure the remaining parameters. If we want to generate a table just for the car type $city$, this could be performed on the basis of the query $P_{[CF \cup \{type=city\}]}S$.

(3) “Local Consistency”. An alternative to the previously discussed knowledge representations is to use tables that represent local consistency properties. For example, the constraint $c_1 : 4wheel = yes \rightarrow type = xdrive$ can be represented by a corresponding *consistency table* (variant table [7]) c_1 expressing all possible combinations of variable values of $4wheel$ and $type$ as specified by the corresponding constraint c_1 (see Table 5).

Table 5

Implicit configuration space description representing locally consistent variable value combinations (Table $c1$) – in this case, combinations specified by $c_1 : 4wheel = yes \rightarrow type = xdrive$.

4wheel	type
yes	xdrive
no	xdrive
no	city
no	limo
no	combi

In a similar fashion, we can define a consistency table c_2 expressing the possible variable value combinations as defined by constraint c_2 (see Table 6). This procedure needs to be performed for each constraint $c_i \in CF$.

This way, we are able to specify tables fulfilling the property of arc consistency since only variable values are included which are part of at least one tuple included in the corresponding consistency table. Following this knowledge representation, the database query in our car configuration task is the following (see Query 4).

4. SELECT * FROM c_1 , c_2 , ...
WHERE $c_1.type = c_2.type$ AND ..

Table 6

Implicit configuration space description representing locally consistent variable value combinations (Table c2) – in this case, combinations specified by $c_2 : \text{skibag} = \text{yes} \rightarrow \text{type} \neq \text{city}$.

skibag	type
yes	xdrive
yes	limo
yes	combi
no	xdrive
no	limo
no	combi
no	city

In this setting, we again assume that S includes all variables. Furthermore, P can be regarded as the table related to the *equi-join* of all generated consistency tables, i.e., $c_1 \bowtie c_2 \bowtie c_3 \bowtie c_4 \bowtie c_5$ in our case where table c_i represents the corresponding constraint $c_i \in CF$. In this context, join conditions have to be integrated in the query for every combination of consistency tables where there is an overlap in terms of the included attributes. For example, consistency tables c_1 and c_2 both include the *type* attribute. Consequently, $4\text{wheel.type} = \text{skibag.type}$ has to be included as join condition into the query. Table 7 shows the complete set of (partial) configurations returned by Query 4 if we assume that only Tables c_1 and c_2 have been defined and included into the query.

Table 7

Partial configurations determined by Query 4.

4wheel	type	skibag
yes	xdrive	yes
yes	xdrive	no
no	xdrive	yes
no	xdrive	no
no	city	no
no	limo	yes
no	limo	no
no	combi	yes
no	combi	no

Example Query Optimization. Following the idea of k -consistency in constraint-based reasoning [1], the number of tuples in a consistency table can be further reduced. For example, if the car type *city* is included in Table c_1 but there does not exist a tuple in c_2 with *type* = *city*, all tuples of c_1 including *city* can be removed as well. Formulated differently, we could check each entry of each consistency table for the existence of a solution which can lead to a further reduction of the number of tuples in the existing consistency tables. Following this idea, we are able to guarantee global consistency [1] meaning that each consistency table only includes tuples which are part of at least one configuration. Furthermore, as a result of related work [8], the inclusion of negative consistency

tables and/or tables representing conflicts could make sense to further improve database query efficiency.

4. Initial Performance Analysis

We compared the performance of the three discussed approaches for representing configuration tasks as database query with constraint solving on the basis of five real-world feature models [9, 10] selected from the S.P.L.O.T. feature model repository [11]. Table 8 provides an overview of selected feature models. Due to space complexity, not all configurations could be determined for *TTax* and *FQAs* within reasonable time limits.

For each feature model, we randomly synthesized² and collected 25,000 user requirements that cover 40% of the leaf features in the feature model. We applied the systematic sampling technique [12] to select 10 *no-solution* user requirements and 10 user requirements with at least one solution. In Table 9, each setting shows the average runtime of the corresponding approach after executing the queries on the basis of these 20 user requirements. We used Choco Solver³ and HSQLDB⁴ as an in-memory relational database management system. All experiments were run with an Apple M1 Pro (8 cores) with 16-GB RAM, and an HSQLDB maximum cache size of 4GB.

Table 9 shows the results of this evaluation of selected feature models represented as (1) an explicit enumeration of *all possible configurations*, (2) an implicit representation of the feature model configuration space (*one table per variable*), (3) an implicit representation where individual tables represent *local consistency*, and (4) constraint satisfaction problem (CSP). Corresponding evaluation results show similar runtimes for small models and significantly longer runtimes for more complex models. Basically, the results of our performance evaluation show the applicability of database query based configuration approaches.

Table 8

Feature models used for evaluation purposes (IDE=IDE product line, DVS=digital video system, DELL=DELL Laptops, MTT=Model transformation taxonomy, FQAs=Functional Quality Attributes, FM=feature model, F= features, LF=leaf features, HC=hierarchical constraints, CC = cross-tree constraints, and CONFS = configurations).

FM	IDE	DVS	DELL	MTT	FQAs
#F	14	26	47	88	178
#LF	9	16	38	55	124
#HC	11	25	16	54	92
#CC	2	3	105	0	9
#CONFS	80	22,680	2,319	-	-

²To ensure the reproducibility of the results, we used the seed value of 141982L for the random number generator.

³choco-solver.org

⁴hsqldb.org

Table 9

The average runtime (*msec*) of database query and constraint-based configuration (FM = feature model, ALLC = all configurations (no optimization), OTV = one table/variable (with node consistency), OTC = one table/constraint (with arc consistency), and CSP = constraint satisfaction problem).

FM	IDE	DVS	DELL	MTT	FQAs
ALLC	0.05	3.66	1.44	-	-
OTV	0.49	0.45	2.53	1,448	301,541
OTC	0.51	0.78	3.32	704	220,922
CSP	0.73	0.78	1.09	1.19	2.43

5. Threats to Validity

We have shown how to apply database queries to the identification of configurations. In a performance analysis, we compared the runtimes of database queries with the Choco constraint solver. Related results show the basic applicability of our approach, however, further evaluations and optimizations are needed – specifically with industrial datasets. Our focus in this paper is a discussion of basic alternative knowledge representation approaches that can be used as a basis for defining database queries. We are aware of related work focusing on compression aspects when supporting configuration with variant tables – see, for example, Haag [7]. A major issue for our future work will be to understand the possibilities of knowledge compression depending on the used knowledge representation. Finally, integrating machine learning with constraint solving is a relevant topic [13] – a major goal for future work is to analyze related application potentials in the context of database queries [14].

6. Conclusions and Future Work

We have introduced a database query based approach to constraint-based configuration. With this, we provide an alternative to approaches such as SAT solving and constraint solving. For sure, further evaluations need to be performed and the proposed queries have to be further optimized for more complex industrial scenarios.

Open issues for future related work are the following: (1) further evaluations on the basis of industrial configuration benchmarks, (2) comparison with other knowledge representation and reasoning approaches such as answer set programming (ASP) and SAT solving, (3) performance improvements through parallelization approaches (e.g., [15]), (4) understanding in more detail how constraint-based reasoning and database queries can profit from each other, for example, in which way could forward checking be applied in database queries and in which way can techniques from relational databases be useful in the context of SAT and constraint solving, and (5) we are interested in which way machine learning and knowl-

edge compression can be used and combined to increase database query efficiency.

References

- [1] F. Rossi, P. van Beek, T. Walsh, *Handbook of Constraint Programming*, Elsevier, 2006.
- [2] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, *Knowledge-based Configuration - From Research to Business Cases*, Elsevier, 2014.
- [3] U. Junker, Configuration, in: F. Rossi, P. van Beek, T. Walsh (Eds.), *Handbook of Constraint Programming*, Elsevier, 2006, pp. 837–873.
- [4] J. Gu, P. W. Purdom, J. Franco, B. W. Wah, Algorithms for the Satisfiability (SAT) Problem: A Survey, in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1996, pp. 19–152.
- [5] V. Myllärniemi, J. Tiihonen, M. Raatikainen, A. Felfernig, Using Answer Set Programming for Feature Model Representation and Configuration, in: ConfWS’14, Novi Sad, Serbia, 2014, pp. 1–8.
- [6] U. Junker, QuickXPlain: Preferred Explanations and Relaxations for Over-constrained Problems, in: AAAI 2004, 2004, pp. 167–172.
- [7] A. Haag, Managing Variants of a Personalized Product, *JIIS* 49 (2017) 59–86.
- [8] A. Haag, Arc Consistency with Negative Variant Tables, in: ConfWS’15, Vienna, Austria, 2015, pp. 81–87.
- [9] D. Benavides, S. Segura, A. Ruiz-Cortes, Automated Analysis of Feature Models 20 Years Later: A Literature Review, *Inf. Sys.* 35 (2010) 615–636.
- [10] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-oriented Domain Analysis (FODA) – Feasibility Study, Technical Report SEI-90-TR-21 (1990).
- [11] M. Mendonca, M. Branco, D. Cowan, S.P.L.O.T.: Software Product Lines Online Tools, in: OOPSLA ’09, ACM, New York, NY, USA, 2009, pp. 761–762.
- [12] S. A. Mostafa, I. A. Ahmad, Recent Developments in Systematic Sampling: A Review, *Journal of Statistical Theory and Practice* 12 (2018) 290–310.
- [13] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V. Le, K. Pilsl, M. Enzelsberger, T. Tran, An Overview of Machine Learning Techniques in Constraint Solving, *JIIS* 58 (2022) 91–118.
- [14] R. Guo, K. Daudjee, Research Challenges in Deep Reinforcement Learning-Based Join Query Optimization, in: aiDM’20, ACM, New York, USA, 2020.
- [15] V. Le and C. Vidal Silva and A. Felfernig and D. Benavides and J. Galindo and T.N.T. Tran, FastDiagP: An Algorithm for Parallelized Direct Diagnosis, in: AAAI-23, 2023, pp. 6442–6449.

Game-based Configuration Task Learning with ConGUESS: An Initial Empirical Analysis

Andreas Hofbauer^{1,*}, Alexander Felfernig¹

¹Graz University of Technology, Inffeldgasse 16b, Graz, 8010, Austria

Abstract

The concepts and semantics of constraint solving and configuration need to be understood in order to be able to develop one's own configuration knowledge bases. Developing a related basic understanding is in many cases quite challenging. Consequently, further support is needed that makes the learning of configuration knowledge representation practices and semantics less effortful. In this paper, we provide a short overview of ConGUESS which is a game-based learning environment for constraint-based configuration tasks. In this context, we report the results of a user study which focused on an analysis of the perceived complexity of different constraint types and on a corresponding usability analysis.

Keywords

Knowledge-based Configuration, Constraint Solving, E-Learning, Gamification

1. Introduction

Assuring the correct understanding of configuration knowledge representations and corresponding semantics is an important issue specifically in industrial configuration settings. Such an understanding can be regarded as a precondition for successful configurator development and maintenance [1, 2, 3]. Following the basic idea of gamification-based learning [4], we have developed ConGUESS [5] which is an application supporting the learning of the semantics of constraint satisfaction problems (CSP) [6] in a gamification-based fashion.

The overall idea of ConGUESS is to pre-generate configuration tasks (represented as CSPs) and let users (game players) try to figure out correct solutions for the defined tasks. With this, ConGUESS follows the idea of earlier related work focusing on the learning of graphical configuration constraints (specifically, incompatibility constraints) and the concepts of hitting sets in model-based diagnosis (specifically, minimal food item sets that cover all relevant vitamins) [7, 8, 9].

Also in this line of research, Jefferson et al. [10] present the application COMBINATION which supports the learning of configuring color ray emitting wooden pieces such that no color array hits a wooden piece of different color. Compared to related work, ConGUESS extends the expressivity of constraint representations and also includes

a gamification-based approach that can help to increase user engagement.

The contributions of this paper are the following: (1) we provide a short introduction to the ConGUESS gaming app, (2) we report the results of an initial complexity and usability analysis that has been conducted in an Artificial Intelligence university course, and (3) we discuss different open issues for further related research.

The remainder of this paper is organized as follows. In Section 2, we provide a short overview of the ConGUESS app specifically introducing the major idea behind. Thereafter, in Section 3, we discuss first insights regarding the perceived complexity of different constraint types. In Section 4, we report results regarding the usability of ConGUESS. In Section 5, we discuss potential threats to validity. The paper is concluded with a discussion of open research issues in Section 6.

2. The ConGUESS Game

In the line of related research (e.g., [7]), ConGUESS is provided as Android app¹ which includes mechanisms for automated CSP generation and evaluation of solutions.² In ConGUESS, players have to solve pre-generated CSPs [6] which are represented in terms of a set of Variables V with related domain definitions and a corresponding set of constraints (C). The task of players is to identify solutions (configurations) that satisfy all given constraints.

ConGUESS supports different game levels where with an increasing level the corresponding CSPs become more difficult to solve. For solving configuration tasks (CSPs), players (users) have a pre-defined time limit. For each correctly solved CSP, players receive corresponding points

¹See play.google.com.

²We apply the constraint solving library Choco (choco-solver.org).

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

✉ andreas.hofbauer@student.tugraz.at (A. Hofbauer);

alexander.felfernig@ist.tugraz.at (A. Felfernig)

🌐 https://felfernig.ist.tugraz.at/ (A. Felfernig)

👤 0000-0003-2956-3862 (A. Hofbauer); 0000-0003-0108-3146

(A. Felfernig)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

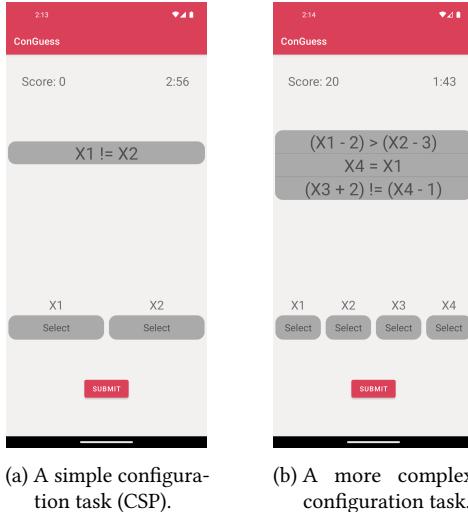


Figure 1: ConGUESS: configuration task user interface.

which increases their overall game score. If a player proposes a configuration *inconsistent* with the given set of constraints, his/her score is not reduced and further tries are possible. With an increasing number of unsuccessful tries, the number of points that can be received for a correct solution gets decreased. Finally, the game provides a global highscore ranking which helps to further motivate users to improve their personal highscore.

A screenshot of ConGUESS in action is provided in Figure 1 which depicts two different configuration tasks (a more simple one on the left hand side and a more complex one on the right hand side). The value of each corresponding variable has to be specified individually indicated by the *select* button. A major objective of the app is to make the configuration task representation as understandable as possible. For this reason, the overall rule of the app in terms of information visualization is that each configuration task fits into the screen without the need of scrolling.

As mentioned, constraint satisfaction problems (CSPs) in ConGUESS are pre-generated. The consistency of individual configuration tasks (CSPs) is checked with the CHOCO constraint solver. If a generated configuration task (variables and corresponding constraints) is consistent, the corresponding setting is stored for further usage (as configuration task given to players). Player-proposed solutions as well as generated configuration tasks are checked for consistency using CHOCO.³

³Details on the ConGUESS constraint solving and configuration task generation approach can be found in Hofbauer and Felfernig [5].

3. Complexity of Constraint Types

Our goal was to better understand in which way different types of constraints are understood by players. In order to achieve this goal, we performed a user study with 150 bachelor students engaged in an Artificial Intelligence course at the Graz University of Technology. Best-performing students had the chance to achieve additional bonus points considered then as a part of the overall evaluation. In total, 780 game sessions have been completed within the scope of the user study resulting in an average number of 5.2 gaming sessions per study participant (with an average of 6 levels per session).

In each ConGUESS session, correct and wrong guesses were tracked in combination with the corresponding configuration task shown to the player. The error rate R_{error} of specific configuration tasks (CSPs) was tracked following the metric shown in Formula 1. In this context, n_{total} is the total amount of guesses and n_{error} is the amount of wrong guesses for a CSP.

$$R_{error} = \frac{n_{error}}{n_{total}} \quad (1)$$

Within the scope of our study, we compared different *configuration task types* with regard to their understandability: configuration tasks (1) consisting of *equality and inequality* constraints, (2) consisting of constraints including a *range restriction*, i.e., $<$, $>$, \leq , \geq , (3) with *implications (requires)* and *equivalences*, and (4) with different *numbers of constraints and variables*.

In a first step, we focused on the analysis of single-constraint configuration tasks, i.e., configuration tasks with only one constraint included ($|C| = 1$). Tables 1–5 include example constraints which represent a cor-

responding analysis class, for example, the constraint $X1 = X2$ in Table 1 represents a configuration task with a singleton constraint of type equality constraint. Similarly, the first constraint in Table 2 represents a configuration task with a single constraint of type $<$.

Equality and Inequality Constraints. First, we have analyzed player failure rates when being confronted with singleton equality and inequality constraints. The corresponding R_{error} rates are depicted in Table 1. As can be immediately seen, the error rates for such constraints are rather low (on an average, below 5%) indicating a high degree of understandability in the reported basic setting.

example constraint(s)	avg. R_{error} in %
$X1 = X2$	4.08
$X3 != X2$	2.11

Table 1
Error rates of binary (in-)equality constraints.

Range Restriction Constraints. In the next step, we analyzed the understandability of range restriction constraints ($<$, $>$, \leq , \geq) (see Table 2). Compared to settings including the $<$ and $>$ operators, error rates significantly increase with settings including \leq and \geq operators. One way to explain this significant difference is the increased complexity of $\{\leq, \geq\}$ due to the fact that both dimensions, inequality and equality have to be taken into account at the same time.

example constraint(s)	avg. R_{error} in %
$X3 < X4$	8.41
$X1 > X2$	4.90
$(X2 + 1) > X3$	7.37
$X3 \leq X2$	20.49
$X2 \geq X1$	13.50

Table 2
Error rates of binary range restriction constraints.

Requires and Equivalence Constraints. In this context, we have compared the understandability of individual implications (requires) and equivalences (see Table 3). We can see that equivalence constraints have slightly lower error-rates than requires constraints. This is a result that has also been confirmed by a previous study of Felfernig et al [3]. One way to explain this difference is a potentially higher overhead induced by the analysis of implications since equivalences can be reduced to settings where both sides of the logical operator must have the same logical value. Further related work on model understandability can be found in Sepasi et al. [11] where cognitive complexity is also measured on the basis of eye tracking technologies.

example constraint(s)	avg. R_{error} in %
$(X2 = X4) \leftrightarrow (X1 < X3)$	8.05
$(X1 > X3) \leftrightarrow (X1 < X2)$	17.39
$(X3 != X2) \rightarrow (X4 != X2)$	18.06
$(X1 > X3) \rightarrow (X1 < X2)$	20.53

Table 3
Error rates of requires and equivalence constraints.

Number of Constraints. When using $\{\rightarrow, \leftarrow\}$ instead of \leftrightarrow for expressing equivalence knowledge, we need twice the amount of constraints. As could be observed in our analysis, an increasing number of constraints leads to increasing error rates due to a lower understandability of the configuration task (see Table 4).

example constraint(s)	avg. R_{error} in %
c1: $X1 > X2$	4.90
c2: $X2 = X3$	
c1: $X1 \geq X2$	
c2: $X1 > X4$	17.12
c2: $X3 > X2$	
c1: $X1 != X3$	
c2: $X2 > X3$	
c3: $X4 \leq X1$	36.87
c4: $X3 < X4$	

Table 4
Error rates with an increasing number of constraints.

N-ary Constraints. The highest error-rates in our study were encountered with constraints involving more than 2 variables. As we can see in Table 5, even configuration tasks with $|C| = 1$ are already difficult to solve for players, if the number of included variables is greater than 2. Furthermore, by increasing the number of constraints in a configuration task, it becomes extremely challenging for players to find a consistent solution.

example constraint(s)	avg. R_{error} in %
c1: $X3 \leq (X1 + 3) = (X2 - 2)$	20.13
c1: $X3 \leq (X4 + 4)$	
c2: $X4 < X1 \leq X3$	72.12
c1: $X3 \geq X1 \leq X2$	
c2: $(X1 \cdot 2) > X2$	73.02
c3: $X3 < X2$	
c1: $X4 != (X3 + 3) < X1$	
c2: $(X4 - 2) \geq X1$	
c3: $X1 < (X3 \cdot 3)$	
c4: $X4 != (X1 + 3)$	84.68

Table 5
Error rates with n-ary constraints.

4. Usability of CONGUESS

To evaluate the usability of CONGUESS, we have performed a usability study with 10 participants (computer science students on the bachelor level). For this purpose, we have used the SYSTEM USABILITY SCALE (SUS) [12]. SUS is a widely used tool for evaluating the usability of systems and software applications and it consists of a questionnaire with 10 items. After using CONGUESS (without further explanations), the participants had to rate their perception of the software on a 5-point *Likert scale*. Following the SUS calculation scheme resulted in an overall evaluation of 89.5 out of 100 which is an excellent SUS rating expressing clear understandability and high willingness to use the system.

5. Threats to Validity

The results reported in this paper are based on the app usage by bachelor-level students within an Artificial Intelligence course. On the one hand, different educational backgrounds could be expected by persons working in industrial configurator projects. On the other hand, persons in an early phase of their career could be in a similar situation as students engaged in our study.

The settings analyzed in our study are limited in the sense that further aspects such as constraint grouping, i.e., constraint ordering, have not been analyzed in detail. Furthermore, we did not compare alternative ways of increasing the complexity level of individual configuration tasks which is important since our goal is to create a kind a flow where game players try to solve even more configuration tasks. We regard these aspects as major issues for future work.

6. Conclusion and Future Work

In this paper, we have presented CONGUESS which is an Android app supporting the learning of configuration knowledge representations and underlying semantics. The basic idea of CONGUESS is that players learn constraint semantics on the basis of trying to solve CSPs. We conducted an empirical study to better understand the cognitive complexity of different constraint structures.

Major issues for future work are the following: (1) we will analyze to which extent the grouping of constraints and corresponding variables can have an impact on the overall understandability of a configuration task, (2) based on the insights of our empirical study, we will propose adaption operations on real-world configuration knowledge bases and analyze related impacts on understandability (again, within the scope of empirical studies), (3) based on the results of further empirical studies,

we will try to adapt the complexity measures currently integrated into the CONGUESS configuration task generation. Improved complexity measures could result in a more user-centered increase of configuration task complexity and with this potentially also to a corresponding improved learning experience.

References

- [1] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration - From Research to Business Cases, Elsevier, 2014.
- [2] A. Felfernig, S. Reiterer, M. Stettinger, J. Tiihonen, Towards understanding cognitive aspects of configuration knowledge formalization, in: Vamos'2015, ACM, New York, NY, USA, 2015, p. 117–123.
- [3] A. Felfernig, M. Mandl, A. Pum, M. Schubert, Empirical knowledge engineering: Cognitive aspects in the development of constraint-based recommenders, in: Trends in Applied Intelligent Systems, volume 6096 of LNCS, Springer, Berlin / Heidelberg, 2010, pp. 631–640.
- [4] R. Raymer, Gamification: Using game mechanics to enhance elearning, ELearn 2011 (2011).
- [5] A. Hofbauer, A. Felfernig, CONGUESS: A Learning Environment for Configuration Tasks, in: ACM SPLC'22, Association for Computing Machinery, New York, NY, USA, 2022, p. 156–157.
- [6] F. Rossi, P. van Beek, T. Walsh, Handbook of Constraint Programming, Elsevier, Amsterdam, The Netherlands, 2006.
- [7] A. Felfernig, M. Jeran, T. Ruprechter, A. Ziller, S. Reiterer, M. Stettinger, Learning games for configuration and diagnosis tasks, in: 17th International Configuration Workshop, volume 1453, CEUR, Vienna, Austria, 2015, pp. 111–114.
- [8] A. Felfernig, M. Schubert, C. Zehentner, An efficient diagnosis algorithm for inconsistent constraint sets, AIEDAM 26 (2012) 53–62.
- [9] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1987) 57–95.
- [10] C. Jefferson, W. Moncur, K. Petrie, Combination: Automated generation of puzzles with constraints, in: ACM Symposium on Applied Computing, ACM, New York, NY, USA, 2011, pp. 907–912.
- [11] E. Sepasi, K. Balouchi, J. Mercier, R. Lopez-Herrejon, Towards a cognitive model of feature model comprehension: An exploratory study using eye-tracking, in: ACM SPLC'22, Association for Computing Machinery, New York, NY, USA, 2022, p. 21–31. URL: <https://doi.org/10.1145/3546932.3546995>. doi:10.1145/3546932.3546995.
- [12] J. Brooke, SUS: A quick and dirty usability scale, Usability Eval. Ind. 189 (1995).

Collaborative Recommendation of Search Heuristics For Constraint Solvers*

Damian Garber^{1,*}, Tamim Burgstaller¹, Alexander Felfernig¹, Viet-Man Le¹, Sebastian Lubos¹, Trang Tran¹ and Seda Polat-Erdeniz¹

¹Graz University of Technology, Inffeldgasse 16b, Graz, 8010, Austria

Abstract

Feature models (FM) support the management of variability properties of software, products, and services. To enable feature model configuration, these models have to be translated into a corresponding formal representation (e.g., a satisfiability or constraint satisfaction representation). Specifically in interactive configuration, efficient response times are crucial. In this paper, we show how to improve the performance of constraint solvers (supporting FM configuration) on the basis of exploiting the concepts of collaborative filtering for recommending solver search heuristics (variable orderings and value orderings). As a basis for our recommendation approach, we used data (configurations) synthesized from real-world feature models using different state-of-the-art synthesis approaches. A performance analysis shows that, with heuristics recommendation, significant improvements of solver runtime performance compared to standard solver heuristics can be achieved.

Keywords

Feature models, configuration, constraint solving, search heuristics, performance optimization, collaborative filtering

1. Introduction

Feature models (FMs) are in wide-spread use for modeling variability properties [1, 2]. These properties can be translated into a formal representation [3] to support various types of reasoning tasks, for example, in the context of feature model analysis and feature model configuration. In this paper, we focus on the aspect of *feature model configuration* where users of a configuration system define their preferences (e.g., in terms of intended feature inclusions) and the feature model configurator then tries to find a corresponding complete configuration which defines inclusion or exclusion for each feature.

Feature model configuration needs to be efficient which can become challenging specifically with large and complex configuration knowledge bases. The major means of improving the performance of solvers (specifically SAT and constraint solvers) is to employ different *search heuristics* which can help to cut down the search space as fast as possible. Following the idea of integrating machine learning (ML) with constraint solving [4], we propose to apply recommender systems [5], more specifically, collaborative filtering [6], to recommend solver

search heuristics for a new FM configuration task.

A major precondition for implementing such a machine learning approach is the availability of training data that help to identify relevant heuristics. An issue in this context is the availability of datasets – this cannot be guaranteed specifically at the very beginning when an FM configurator has not been used that often. An alternative to datasets collected from real-world user interactions is data synthesis [7, 8]. In this paper, we focus on developing and comparing different configuration data synthesis strategies (see, e.g., Pereira et al. [8]) to gain deeper insights regarding the impact of the used strategies on the quality of the heuristics determined by our collaborative filtering approach. As discussed in Pereira et al. [8] (the focus of their work is performance prediction, for example, in video encoding scenarios), performance prediction is feasible, however, synthesizing high-quality and small sample data is a challenging task (see also [9]).

There exist a couple of approaches to integrate machine learning with constraint solving focusing on the aspect of identifying relevant heuristics on the basis of given datasets – see, for example, Erdeniz et al. [10] and Uta et al. [11]. These approaches focus on *predicting* relevant attribute values (features) for a user and – at the same time – *improving* constraint solver performance by choosing appropriate *variable value ordering heuristics*. In contrast to related work, we extend the recommendation scope by also supporting the identification of efficient *variable orderings*. At the same time, we analyze the impact of different data synthesis strategies on the quality of the recommended heuristics (which we regard as a new contribution to the fields of feature modeling and knowledge-based configuration).

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

✉ dgarber@ist.tugraz.at (D. Garber); tamim.burgstaller@ist.tugraz.at (T. Burgstaller); alexander.felfernig@ist.tugraz.at (A. Felfernig); vietman.le@ist.tugraz.at (V. Le); slubos@ist.tugraz.at (S. Lubos); ttrang@ist.tugraz.at (T. Tran); spolater@ist.tugraz.at (S. Polat-Erdeniz)

DOI 0009-0005-0993-0911 (D. Garber)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License

Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

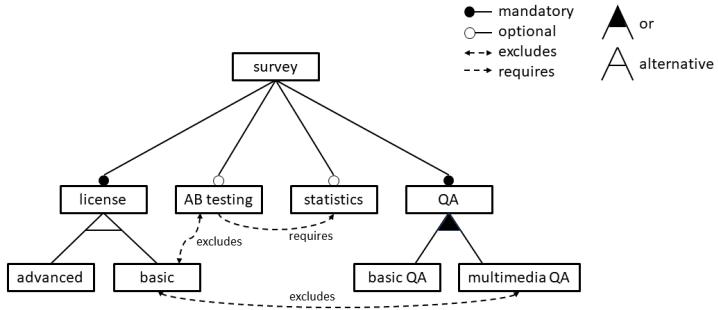


Figure 1: Example feature model of a *survey software* (based on Le et al. [12]).

The major contributions of this paper are the following. (1) we show how to apply configuration space learning concepts in FM configurator performance optimization. (2) our recommendation approach takes into account both, the recommendation of variable orderings and variable value orderings. (3) we compare different data synthesis strategies with regard to their applicability in search heuristics selection. (4) our evaluation results on the basis of real-world configuration (feature) models [13, 14] indicate significant performance improvements.

The remainder of this paper is organized as follows. In Section 2, we provide an example feature model with the related constraint-based representation. In Section 3, we introduce an approach to collaborative filtering based recommendation of constraint solver search heuristics. In Section 4, we provide an overview of the synthesis approaches we have used in our recommendation settings. Performance evaluation results are summarized in Section 5. Threats to validity are discussed in Section 6. The paper is concluded with a discussion of future research issues in Section 7.

2. Example Configuration Task

In the following, we introduce an example feature model representing the variability properties of a *survey software* (see Figure 1). Each configured *survey* software must have included a corresponding *license* model (which can be either *advanced* or *basic*). The features *statistics* and *ABtesting* are optional ones, i.e., must not be part of every configuration. Finally, each survey software configuration must include a selected interaction mode (in terms of the type of questions (feature *QA*) supported) which consists of at least one out of question answering (feature *basicQA*) and multimedia based question answering (feature *multimediaQA*).

The feature model in Figure 1 includes different relationships and cross-tree constraints $c_i \in C$. First,

each survey software configuration must include the root feature ($c_1 : \text{survey} = \text{true}$) and either an advanced or basic license ($c_2 : \text{survey} \leftrightarrow \text{license}$ and $c_3 : \text{license} \leftrightarrow \text{advanced} \vee \text{basic}$).¹ Furthermore, ABTesting and statistics are optional ($c_4 : \text{ABtesting} \rightarrow \text{survey}$ and $c_5 : \text{statistics} \rightarrow \text{survey}$). Each selected question mode must include at least one out of basic and multimedia ($c_6 : \text{QA} \leftrightarrow \text{survey}$ and $c_7 : \text{QA} \leftrightarrow \text{basicQA} \vee \text{multimediaQA}$). Finally, the FM includes a set of cross-tree constraints: basic licenses are incompatible with ABTesting ($c_8 : \neg(\text{basic} \wedge \text{ABtesting})$), the inclusion of ABTesting requires the inclusion of statistics ($c_9 : \text{ABtesting} \rightarrow \text{statistics}$), and a basic license must not be combined with a multimedia answering mode ($c_{10} : \neg(\text{basic} \wedge \text{multimediaQA})$).

Summarizing, our example feature model includes the list of (Boolean-valued) features (variables v_i) $F = \{v_1 : \text{survey}, v_2 : \text{license}, v_3 : \text{ABtesting}, v_4 : \text{statistics}, v_5 : \text{QA}, v_6 : \text{basic}, v_7 : \text{advanced}, v_8 : \text{basicQA}, v_9 : \text{multimediaQA}\}$. Furthermore, the model includes the set of constraints $C = \{c_1..c_{10}\}$. These are the two major elements of an *FM configuration task* defined in terms of a constraint satisfaction problem (CSP) (see Definition 1).

Definition 1 (FM Configuration Task). An FM configuration task (V, C, R) can be defined as a CSP, where V is a set of (Boolean-valued) variables $V = \{v_1, \dots, v_n\}$ and $C = \{c_1..c_m\}$ is a set of feature model constraints. Finally, $R = \{r_1..r_k\}$ is a set of user requirements also represented in terms of constraints (mostly variable assignments).

With a configuration task definition², we can introduce the concept of an *FM configuration* (Definition 2).

Definition 2 (FM Configuration). An FM configuration for an FM configuration task (V, C, R) is an assign-

¹ \vee denotes a logical xor.

²Without loss of generality, we focus on feature models and corresponding Boolean variable domains.

Table 1

Solver search heuristics for solving a new configuration task (*task*) can be determined by reusing the search heuristics already applied to create the nearest neighbor (*NN*) configuration(s) (*id* = configuration identifier). In this example, the selected nearest neighbor is configuration 3 – the corresponding search heuristics can be applied to solve the new FM configuration task. In this context, H= *highest value first* and L= *lowest value first*.

id	Configuration								Variable Ordering								Value Ordering								Runtime [ms]		
	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8			
$conf_1$	1	1	0	1	1	1	0	1	9	3	6	1	8	4	7	5	2	L	L	H	L	L	H	H	L	229.133	
$conf_2$	1	1	0	1	1	1	0	1	0	5	2	3	9	7	1	8	6	4	H	L	L	L	H	L	L	218.384	
$conf_3$	1	1	0	1	1	0	1	1	4	2	7	5	8	6	1	3	9	H	H	H	H	L	H	L	L	191.296	
$conf_4$	1	1	0	0	1	0	1	1	0	8	2	7	9	6	5	4	1	3	H	L	H	H	L	H	L	116.995	
task	?	1	?	1	?	?	?	?	?	?	?	?	?	?	?	?	?	?	H	H	H	H	L	H	L	L	?

ment $conf = \{v_1 = va_1 \wedge \dots \wedge v_n = va_n\}$ where $conf \cup C \cup R$ is consistent and every variable in V has an assignment, i.e., we assume assignment completeness.

Assuming a set of defined user requirements $R = \{r_1 : advanced = true, r_2 : ABtesting = true, r_3 : basicQA = true\}$ (i.e., users do not need to define their preferences with regard to all features) could result in the following complete configuration $conf = \{survey = true, license = true, advanced = true, basic = false, ABtesting = true, statistics = true, QA = true, basicQA = true, multimediaQA = false\}$.

Having introduced the concepts of a configuration task and a corresponding configuration, we are now able to discuss our collaborative filtering based constraint solver search heuristics recommendation approach in more detail.

3. Collaborative Search Heuristics Recommendation

Our basic idea is to apply different types of nearest neighbor based collaborative filtering [6] for the purpose of recommending relevant solver search heuristics (including both, *variable orderings* (the order in which the solvers tries to instantiate variables) and *variable value orderings* also denoted as *variable domain strategies* (the order in which the solver instantiates variable values) for a new configuration task (see Definition 1).

Our used variable value orderings (i.e., variable domain strategies) are *highest first* (H) and *lowest first* (L), i.e., the constraint solver starts with trying to instantiating the highest or the lowest variable value first.³ For the purposes of our experiments, we use different data synthesis approaches [9] (see Section 4). Each entry of a synthesized dataset represents a complete configuration consistent with the constraints in C and R (see Definition 2). In addition to the feature settings (inclusion or exclusion), each entry also includes information about (1) the used variable ordering, (2) variable value ordering, and

³In our example, strategy *H* first tries to include feature v_i , i.e., $v_i = true$.

(3) runtime (in ms) to find the corresponding configuration (see Table 1). We use such entries to identify (reuse) search heuristics for completing new configuration tasks.

k-nearest neighbors (k=1). Table 1 shows a simplified example of how to apply *kNN* (*k* nearest neighbor) based approaches for recommending search heuristics for a new configuration task (in this example, we assume $k = 1$). The table contains four (in our case synthesized) entries of complete configurations including further information on the used solver search heuristics, i.e., variable and variable value orderings. Finally, for each configuration we have information about the corresponding solver runtime (in ms).

In this example, the new configuration *task* needs to be solved. The idea is to identify nearest neighbor configurations $conf_i$ on the basis of the similarity between the new configuration task and the available (complete) configuration entries ($conf_1 - conf_4$ in Table 1). Following Formula 1 for determining the similarity between the new configuration task and each $conf_i$, configurations $conf_1 - conf_3$ have the same similarity, i.e., $sim(task, conf_1) = sim(task, conf_2) = sim(task, conf_3) = 1.0$.⁴ In this context, V' denotes variables with associated specified user requirements, i.e., those variables of V which have a corresponding initial value assignment in the configuration *task* definition (e.g., $\{v_2, v_4\}$ in Table 1).

$$sim(task, conf) = \frac{|\{v_i \in V' : val(v_i, task) = val(v_i, conf)\}|}{|\{v_i \in V\}|} \quad (1)$$

In this context, *task* (the initial user requirements) represents a partial configuration, since not every variable needs to have an assigned value – assigned values are assumed to represent user requirements $r_i \in R$.

k-nearest neighbors (k>1). The *k-nearest neighbor* based approach identifies the *k* most similar configurations $conf_i$ compared to the current configuration *task* (see Formula 1) and then chooses the configuration with

⁴ $val(v_i)$ denotes the value of variable v_i .

the best solver runtime performance (which is then the so-called nearest neighbor). Since $conf_3$ has the lowest (best) runtime among the identified nearest neighbors, we can reuse the solver search heuristics used to determine $conf_3$. If $k = 1$, only one nearest neighbor is identified and the corresponding solver search heuristics are applied to the current configuration task. The major difference between $k = 1$ and $k > 1$ is that in $k > 1$ settings it could be the case that a configuration with lower similarity (see Formula 1) is selected due to a better corresponding runtime performance.

k-nearest neighbors ($k>1$, weighted). The previously discussed k-nearest neighbor approach takes into account the similarity between the current configuration (task) and already existing configurations $conf_i$. In our experiments, we were also interested in the impact of taking into account tradeoffs between configuration similarity and solver runtimes (see Formula 2).

$$sim_t(task, conf) = \frac{\max(runtime) - runtime(conf)}{1 - sim(task, conf) + \lambda} \quad (2)$$

Similar to the *k-nearest neighbor* based approach ($k > 1$), this weighted approach (see Formula 2) as well identifies configurations similar to the current configuration (task), but then uses similarity as a weighting factor, i.e., not just selects the nearest neighbor with the best runtime performance. This way, we determine those nearest neighbors with a good runtime which are at the same as similar as possible to the given configuration *task*.

In Formula 2, we have introduced a small constant λ to avoid division by 0 which could happen in situations where the requirements in the given configuration *task* are equivalent with the corresponding variable settings in *conf*, i.e., the similarity is 1.0. Finally, we want to mention that $\max(runtime)$ denotes the highest (global) runtime value used to represent the worst (highest) runtime observed in the (FM-specific) synthesized data.

4. Used Data Synthesis Approaches

Overall Synthesis Approach. The methods we chose for synthesizing configuration data are based on those discussed in Pereira et al. [9]. We have applied these synthesis approaches in our constraint solver search heuristics recommendation scenario for the purpose of generating complete and consistent configurations, i.e., each variable has a corresponding assignment and all assignments are consistent with the constraints in C . As solver input, we have generated a set of user requirements $r_i \in R$ representing around 10% of the features contained in the

corresponding feature model. For each variable, a corresponding variable value ordering heuristics has been chosen randomly. Finally, we also chose a variable ordering for variables not contained in the generated set of user requirements R . On the basis of this initial input (randomly generated search heuristics and user requirements), a solver has been activated with a repetition factor of 5 to determine the average runtime needed to solve the defined setting (see also Tables 2 and 3). Following the overview of Pereira et al. [9], we have used and evaluated the following data synthesis approaches.

Random Sampling. Random Sampling is one of the most widely used methods to synthesize data for configuration problems [15, 16, 17]. There are several variations [9, 13] that can be differentiated with regard to the number generated samples. One of these variations generates a fixed (pre-defined) number of configurations, regardless of the properties of the underlying feature model. The *fixed number* approaches we have tested in the context of our evaluation are: 100, 200, 500, 1000, 2000 and 10000.

In addition to this rather static approach, we have applied other approaches which take into account feature model sizes. *Random N* focuses on generating N random configurations where N equals the number of features in the feature model. There also exist some variations of this approach where the number of generated configurations equals $2N$ (Random $2N$) or $3N$ (Random $3N$). Furthermore, the number of configurations can also be systematically reduced by introducing the synthesis variants *Random $\frac{1}{4}N$* , *Random $\frac{1}{2}N$* and *Random $\frac{3}{4}N$* .

Heuristics Based Sampling. Other approaches are based on heuristics for configuration generation.

Feature Frequency Heuristic (FFH) The Feature Frequency Heuristic (FFH) [18] generates configurations following the strategy of ensuring that each feature occurs at least a predefined times in the resulting configurations (corresponding thresholds can also be defined per feature). In our evaluation, we have applied the (global) feature-wise threshold values of 5, 10, and 20.

Feature Coverage Heuristic (FCH) FCH [19, 20, 18] tries to ensure the presence of every feature combination of size t in the generated data. This approach could be applied in different ways, for example, by generating all possible t -way feature combinations. We used the ACTS tool⁵ provided by [21], which generates so-called covering arrays [22]. This way, we generated data sets with 2-way and with 3-way coverage. Higher coverage comes at very high computational costs for larger models which forced us to omit corresponding synthetizations with our used ACTS tool [21].

⁵<https://csrc.nist.gov/projects/automated-combinatorial-testing-for-software>.

Table 2

Constraint solver performance with k-nearest neighbor based (following the nearest neighbor selection approach of Formula 1) search heuristics recommendations. In this context, the constraint solver performance of different feature models has been evaluated. Compared to standard solver runtimes (without heuristics recommendations – see Table 4)) , we can observe significant corresponding runtime improvements. Values in bold indicate the best configuration synthesis strategy, values with a grey background the best corresponding k value.

	Linux				UCLinux-distribution				Busybox				WeaFQAS				REAL-FM-11				MobileMedia				
	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	
N = 100	366.39	371.05	371.82	370.39	30.20	29.31	29.21	29.23	6.17	5.93	5.92	5.93	0.49	0.36	0.36	0.35	0.17	0.10	0.09	0.13	0.08	0.07	0.07	0.07	
N = 200	371.71	374.69	374.83	372.28	30.13	30.72	30.08	30.08	6.02	6.07	6.21	6.30	0.37	0.36	0.37	0.38	0.07	0.07	0.07	0.07	0.06	0.05	0.05	0.05	
N = 500	378.97	375.05	379.72	377.27	379.48	32.38	32.26	32.28	32.15	7.54	6.52	6.56	6.54	0.41	0.42	0.42	0.42	0.09	0.11	0.12	0.12	0.06	0.06	0.06	0.05
N = 1000	384.09	378.44	377.27	377.27	379.48	30.30	30.45	30.57	30.68	8.73	8.87	9.03	9.07	0.66	0.68	0.67	0.17	0.17	0.17	0.17	0.12	0.12	0.12	0.08	
N = 2000	394.45	393.54	393.54	393.77	390.67	36.60	36.64	36.75	36.96	9.24	8.73	8.87	8.93	0.51	0.52	0.51	0.52	0.11	0.11	0.12	0.12	0.08	0.08	0.08	0.08
N = 10000	500.68	510.78	484.87	510.41	69.44	70.38	69.36	70.00	19.86	21.31	21.54	21.51	3.38	3.57	3.60	0.73	0.75	0.74	0.76	0.53	0.53	0.52	0.53		
Random $\frac{1}{N}$	406.34	387.72	386.52	389.04	31.55	31.79	31.52	31.46	6.17	6.14	6.19	6.12	0.34	0.34	0.34	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
Random $\frac{1}{N}$	419.47	401.34	399.21	405.95	32.17	31.61	32.07	31.77	6.46	6.46	6.46	6.46	0.35	0.35	0.35	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
Random $\frac{1}{N}$	422.74	424.84	426.05	450.48	33.25	33.40	33.32	33.15	6.79	6.73	6.75	6.73	0.35	0.36	0.36	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
Random N	444.35	440.96	438.06	438.72	34.62	34.45	34.79	34.50	6.99	7.03	7.04	7.04	0.36	0.37	0.37	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
Random 2N	525.98	537.85	534.80	513.28	42.60	41.66	40.95	41.43	8.12	8.37	8.48	8.47	0.39	0.40	0.40	0.07	0.07	0.07	0.07	0.04	0.04	0.04	0.04		
Random 3N	632.08	643.38	615.72	635.11	48.20	47.78	48.31	48.44	9.73	9.77	9.90	9.76	0.42	0.42	0.42	0.43	0.07	0.07	0.07	0.07	0.04	0.04	0.04	0.04	
FCH (2-Way)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
FCH (3-Way)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
FFH (5)	386.46	374.25	373.67	374.49	28.68	28.74	28.70	28.69	6.11	5.88	5.92	5.89	0.35	0.35	0.35	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
FFH (10)	372.84	371.83	374.77	378.29	29.14	29.21	29.20	29.14	6.00	5.96	5.94	5.95	0.35	0.36	0.36	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
FFH (20)	377.96	379.57	376.61	380.04	30.17	30.15	30.20	30.06	6.05	6.05	6.07	0.37	0.38	0.38	0.06	0.06	0.06	0.07	0.04	0.04	0.04	0.04			

Table 3

Constraint solver performance with k-nearest neighbor based search heuristics recommendations (following the nearest neighbor selection approach of Formula 2). Again, the constraint solver performance of different feature models has been evaluated. Using this approach, we can observe further solver runtime improvements compared to the basic k-nearest neighbor based approach. Note that for $k = 1$ the performance values are the same as in Table 2 due to the fact that the same heuristics are selected in this case.

	Linux				UCLinux-distribution				Busybox				WeaFQAS				REAL-FM-11				MobileMedia					
	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$	$k = 5$	$k = 10$	$k = 20$		
N = 100	366.39	371.52	385.70	373.99	30.20	29.03	29.10	29.16	6.17	5.96	5.97	5.99	0.49	0.35	0.36	0.36	0.17	0.08	0.07	0.07	0.13	0.06	0.05	0.06		
N = 200	371.71	374.70	376.70	377.71	30.13	30.14	30.10	30.08	6.02	6.06	6.04	6.14	0.37	0.37	0.37	0.36	0.07	0.07	0.07	0.06	0.05	0.05	0.05	0.05		
N = 500	378.97	381.00	377.46	379.25	65.91	30.26	31.76	33.02	6.54	6.50	6.55	6.41	0.42	0.42	0.43	0.42	0.09	0.09	0.09	0.06	0.06	0.06	0.06	0.06		
N = 1000	384.09	381.19	380.19	381.54	32.38	32.37	32.46	32.67	7.54	7.67	7.62	7.69	0.50	0.56	0.50	0.50	0.11	0.11	0.12	0.12	0.08	0.08	0.09	0.09		
N = 2000	394.45	392.21	394.54	388.43	36.60	36.66	36.62	36.57	9.24	8.95	8.74	8.63	0.67	0.66	0.68	0.67	0.17	0.18	0.18	0.12	0.12	0.12	0.13	0.13		
N = 10000	500.68	505.10	514.55	515.44	69.44	70.19	69.13	69.67	19.76	20.44	20.10	19.83	3.53	3.66	3.72	0.73	0.74	0.74	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53
Random $\frac{1}{N}$	406.34	389.73	391.06	391.56	31.55	32.02	31.82	32.04	6.17	6.10	6.11	6.13	0.34	0.34	0.34	0.35	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
Random $\frac{1}{N}$	419.47	405.14	402.24	402.53	32.17	31.87	31.89	31.71	6.46	6.49	6.47	6.49	0.35	0.35	0.36	0.36	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
Random $\frac{1}{N}$	422.74	423.54	426.13	421.33	33.25	33.64	33.39	33.82	6.79	6.74	6.77	6.75	0.35	0.36	0.36	0.37	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
Random N	444.35	442.24	441.59	440.81	34.62	34.34	34.60	35.09	6.99	7.00	7.01	7.01	0.36	0.37	0.37	0.38	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
Random 2N	525.98	528.67	546.50	515.19	42.60	41.43	41.10	42.04	8.12	8.41	8.52	8.60	0.39	0.40	0.40	0.40	0.07	0.07	0.07	0.07	0.04	0.04	0.04	0.05		
Random 3N	632.08	641.79	630.88	636.14	48.20	47.70	47.83	49.73	9.73	9.39	9.47	9.51	0.42	0.43	0.44	0.44	0.07	0.07	0.07	0.07	0.04	0.04	0.04	0.05		
FCH (2-Way)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		
FCH (3-Way)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		
FFH (5)	386.46	374.04	373.87	371.93	28.68	28.76	28.82	28.82	6.11	5.95	6.03	5.98	0.35	0.35	0.35	0.35	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
FFH (10)	372.84	376.19	373.66	374.67	29.14	29.15	29.16	29.28	6.00	5.98	5.99	6.00	0.35	0.36	0.36	0.37	0.06	0.06	0.06	0.06	0.04	0.04	0.04	0.04		
FFH (20)	377.98	377.64	379.47	375.97	30.17	29.88	29.95	30.06	6.04	6.05	6.09	0.37	0.37	0.38	0.38	0.06	0.06	0.06	0.07	0.04	0.04	0.04	0.05			

$$\text{runtime} = \text{time}(\text{NN}) + \text{time}(\text{recommendation}) + \text{time}(\text{solver}) \quad (3)$$

$$\text{runtime} = \text{time}(\text{NN}) + \text{time}(\text{recommendation}) + \text{time}(\text{solver}) \quad (4)$$

Overall, we have compared six different feature models⁶ which significantly differ in terms of the number of variables and the number of corresponding feature model constraints. Tables 2–3 provide an overview of the runtime performance of the used constraint solver Choco⁷ in scenarios where 10% of the user requirements have been specified (randomly assigned) for the new configuration task. To avoid evaluation biases, for each setting (synthesizer strategy \times feature model), we have generated 150 test configuration tasks. In addition, to avoid biases in the training data set, we have generated 5 training datasets for the mentioned settings. With this, we have measured the average runtime needed for solving the configuration task (see also Formulae 3 – 4).

Comparison of Synthesis Strategies. Independent of the used synthesis strategy, solver performance can be

Table 4

Feature models used for evaluating the different heuristics recommendation approaches (also including the standard solver runtime (in *milliseconds*) needed for calculating a solution for a configuration task).

Model	$ V $	$ C $	solver runtime [ms]
Linux	6467	13972	771.93
uClinux-distribution	1580	1793	65.91
busybox	854	905	13.96
WeaFQAS	179	100	1.30
REAL-FM-11	67	64	0.49
mobilemedia	43	32	0.07

significantly improved with search heuristics recommendation, for example, in the context of the Linux feature model (see Table 4), runtimes can be reduced by half. The analysis of the different data synthesis approaches showed that approaches generating smaller datasets in general perform best which can partially explained by the fact that the effort of determining nearest neighbors is reduced. The best performing synthesis approach for small models (*REAL-FM-11 Model* and *MobileMedia Model*) is Feature Coverage Heuristic (2-way) – due to computational overheads, evaluations for larger models have been omitted. For the remaining settings, in the majority of the cases the best performing synthesis approach is the Feature Frequency Heuristic (FFH) with threshold $t = 5$.

Comparison of K-Nearest Neighbor Approaches.

When comparing *k-nearest neighbor* and *weighted k-nearest neighbor* based heuristics recommendation, we can observe that both approaches result in a similar solver performance, however, *weighted k-nearest neighbor* appears to be the more stable approach which is less susceptible to outliers (in terms of low solver performance).

Comparison of k -Values. When comparing different k -values, we can observe a tendency that more complex feature models (and corresponding constraint satisfaction problems) tend to perform better with increasing k -sizes. This can be partially explained by the fact that larger models (with larger configuration/solution spaces) need a higher k -value for achieving a certain coverage of the search space. On the other hand, the additional efforts to be taken into account for increasing k -sizes (e.g., in terms of additional efforts in nearest neighbor determination) can to some extent be compensated by higher-quality variable (value) ordering heuristics.

6. Threats to Validity

The major objective of the presented work is constraint solver optimization, however, the presented approach could also be applied in other application domains such

as operating system optimization and the optimization of production schedules. We regard corresponding evaluations as a major focus of our future work. We are aware of the variety of FM knowledge representations – not all of these representations will directly profit from the concepts presented in this paper (since we focused on specific constraint solver heuristics). On the one hand, we regard related developments, i.e., learning other types of search heuristics, as a major focus of future research. On the other hand, we want to emphasize that the presented collaborative recommendation approaches can be applied as such in other settings with a focus on the reuse of reasoning knowledge. We want to emphasize that we intentionally focused on comparing (memory-based) collaborative recommendation approaches. Future work will include evaluations with model-based (e.g., neural networks) collaborative recommendation approaches. Finally, we are also aware of different types of parallelization approaches helping the improve search efficiency (see, for example, [23, 24]). We regard a direct comparison with such approaches a major task for future work.

7. Conclusions

In this paper, we have presented an approach to recommend constraint solver search heuristics (variable orderings as well as variable values orderings) which help to improve the performance of constraint solver based feature model configuration. We have applied and combined different types of data synthesis strategies and corresponding collaborative recommendation approaches which have been used as a basis for recommending search heuristics for new feature model configuration tasks. The results of our performance evaluation show that an approach to the recommendation of search heuristics combined with a well-fitted data synthesis approach can lead to significant performance improvements in feature configuration (in our evaluation settings, we could observe significant performance improvements of around 50% (and more) compared to standard solver runtimes). Major issues for future work are the evaluation of our approach in further domains (e.g., operating systems optimization) and the development/inclusion of model-based recommendation approaches.

References

- [1] D. Benavides, A. Felfernig, J. Galindo, F. Reinfrank, Automated Analysis in Feature Modelling and Product Configuration, in: ICSR'13, number 7925 in LNCS, Springer, Pisa, Italy, 2013, pp. 160–175.
- [2] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-oriented Domain Analysis (FODA) – Feasi-

- bility Study, Technical Report, SEI, Carnegie Mellon University, Pittsburgh, PA, USA, 1990.
- [3] D. Benavides, P. Trinidad, A. Ruiz-Cortés, Automated reasoning on feature models, in: CAiSE'05, Springer-Verlag, Berlin, Heidelberg, 2005, p. 491–503.
 - [4] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V. Le, K. Pils, M. Enzelsberger, T. Tran, An Overview of Machine Learning Techniques in Constraint Solving, *Journal of Intelligent Information Systems* 58 (2022) 91–118.
 - [5] A. Falkner, A. Felfernig, A. Haag, Recommendation Technologies for Configurable Products, *AI Magazine* 32 (2011) 99–108.
 - [6] M. Ekstrand, J. Riedl, J. Konstan, Collaborative filtering recommender systems, *Found. Trends Hum.-Comput. Interact.* 4 (2011) 81–173.
 - [7] K. Meel, Counting, Sampling, and Synthesis: The Quest for Scalability, in: IJCAI-22, 2022, pp. 5816–5820.
 - [8] J. A. Pereira, M. Acher, H. Martin, J. Jézéquel, Sampling effect on performance prediction of configurable systems: A case study, in: ACM/SPEC International Conference on Performance Engineering, ICPE '20, ACM, 2020, p. 277–288.
 - [9] J. A. Pereira, M. Acher, H. Martin, J. Jézéquel, G. Botterweck, A. Ventresque, Learning software configuration spaces: A systematic literature review, *Journal of Systems and Software* 182 (2021) 111044.
 - [10] S. Polat-Erdeniz, A. Felfernig, R. Samer, M. Atas, Matrix factorization based heuristics for constraint-based recommenders, in: 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), ACM, Limassol, Cyprus, 2019, pp. 1655–1662.
 - [11] M. Uta, A. Felfernig, D. Helic, V. Le, Accuracy- and consistency-aware recommendation of configurations, in: ACM International Systems and Software Product Line Conference (SPLC'22), ACM, Graz, Austria, 2022, pp. 79–84.
 - [12] V. Le, A. Felfernig, M. Uta, T. Tran, C. Vidal, Wipe-OutR: Automated Redundancy Detection for Feature Models, in: 26th ACM International Systems and Software Product Line Conference, ACM, 2022, pp. 164–169.
 - [13] R. Heradio, D. Fernandez-Amoros, J. J. Galindo, D. Benavides, D. Batory, Uniform and scalable sampling of highly configurable systems, *Empirical Software Engineering* 27 (2022) 1–34.
 - [14] M. Mendonca, M. Branco, D. Cowan, S.P.L.O.T.: Software Product Lines Online Tools, in: 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, OOPSLA '09, ACM, 2009, pp. 761–762.
 - [15] M. Acher, P. Temple, J.-M. Jézéquel, J. Galindo, J. Martinez, T. Ziadi, VaryLATEX: Learning Parameter Variants That Meet Constraints, in: 12th International Workshop on Variability Modelling of Software-Intensive Systems, ACM, 2018, p. 83–88.
 - [16] A. Grebhahn, C. Rodrigo, N. Siegmund, F. Gaspar, S. Apel, Performance-influence models of multigrid methods: A case study on triangular grids, *Concurrency and Computation: Practice and Experience* 29 (2017).
 - [17] J. Guo, K. Czarnecki, S. Apel, N. Siegmund, Wałowski, Variability-aware performance prediction: A statistical learning approach, in: 28th IEEE/ACM International Conference on Automated Software Engineering, ASE '13, IEEE Press, Silicon Valley, CA, USA, 2013, pp. 301–311.
 - [18] A. Sarkar, J. Guo, N. Siegmund, S. Apel, K. Czarnecki, Cost-efficient sampling for performance prediction of configurable systems, in: 30th IEEE/ACM International Conference on Automated Software Engineering, ASE '15, IEEE Press, Lincoln, Nebraska, 2015, pp. 342–352.
 - [19] C. Kaltenecker, A. Grebhahn, N. Siegmund, J. Guo, S. Apel, Distance-based sampling of software configuration spaces, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE, Montreal, Quebec, Canada, 2019, pp. 1084–1094.
 - [20] M. Lillack, J. Müller, U. Eisenecker, Improved prediction of non-functional properties in software product lines with domain context, in: S. Kowalewski, B. Rumpe (Eds.), *Software Engineering 2013*, Gesellschaft für Informatik e.V., Bonn, 2013, pp. 185–198.
 - [21] L. Yu, Y. Lei, R. Kacker, D. Kuhn, Acts: A combinatorial test generation tool, in: 6th IEEE International Conference on Software Testing, Verification and Validation, IEEE, Luxembourg, 2013, pp. 370–375.
 - [22] C. Colbourn, Combinatorial Aspects of Covering Arrays, *Le Matematiche* 59 (2004) 125–172.
 - [23] L. Bordeaux, Y. Hamadi, H. Samulowitz, Experiments with Massively Parallel Constraint Solving, in: IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, pp. 443–448.
 - [24] V. Le and C. Vidal Silva and A. Felfernig and D. Benavides and J. Galindo and T.N.T. Tran, FastDiagP: An Algorithm for Parallelized Direct Diagnosis, in: AAAI'23, 2023, pp. 6442–6449.

Solving Multi-Configuration Problems: A Performance Analysis with Choco Solver

Benjamin Ritz^{1,*}, Alexander Felfernig¹, Viet-Man Le¹ and Sebastian Lubos¹

¹Graz University of Technology, Inffeldgasse 16b, 8010 Graz, Austria

Abstract

In many scenarios, configurators support the configuration of a solution that satisfies the preferences of a single user. The concept of *multi-configuration* is based on the idea of configuring a set of configurations. Such a functionality is relevant in scenarios such as the configuration of personalized exams, the configuration of project teams, and the configuration of different trips for individual members of a tourist group (e.g., when visiting a specific city). In this paper, we exemplify the application of multi-configuration for generating individualized exams. We also provide a constraint solver performance analysis which helps to gain some insights into corresponding performance issues.

Keywords

Knowledge-based Configuration, Multi-Configuration, Performance Analysis

1. Introduction

Configuration is the process of assembling basic components into a complex product while taking into account a set of constraints [1, 2, 3, 4]. Most existing configurators are based on the assumption that a solution (configuration) is developed for a single user. On the contrary, group-based configuration [5, 6] focuses on the configuration of a solution for a group of users. Such a configuration must satisfy the preferences of each individual user as much as possible [7, 8]. Group-based configuration can be further extended to allow the configuration of a set of solutions based on the preferences of one or multiple users. Such a *multi-configuration problem* [9] includes a set of constraints specifying restrictions with regard to (1) the combination of multiple solutions and (2) properties of a specific solution. Scenarios including such configuration sets may benefit from multi-configuration. Related example scenarios are the following.

Multi-exam configuration. individual exams are configured for each student, where related constraints are specified by instructors and possibly also students. A configurator can support instructors during the exam preparation phase and helps in the prevention of cheat-

ing through the generation of individualized exams [10].

Project team assignment. persons are assigned to teams such that each team has the expertise to successfully complete the corresponding project (assigned to the team). In this context, fairness aspects can play a role, for example, each team should have at least similar chances to complete a project within pre-defined time limits [9].

Generation of test cases. The automated generation of test cases can be considered as a multi-configuration problem, where input values need to be generated in such a way that given coverage criteria are fulfilled [11].

Holiday planning. Members of tourist groups often do not share the same interests during excursions [12], i.e., which sightseeing destinations to visit. Therefore, a configurator could configure different trips for subgroups of tourists based on their preferences.

Configuration space learning. Many (software) systems (e.g., operating systems) offer a high degree of configurability. In this context, it is difficult to find the optimal configuration settings [13] also due to the fact that it is impossible to test all possible settings to find the optimal one. Multi-configuration can support the identification of test configurations that help to learn dependencies between configuration parameters.

In the context of *multi-exam configuration*, we have built a software library that helps to configure exams. Example inputs are the number of examinees, a pool of questions, and a set of constraints specifying preferences of instructors and examinees. The outcome is a set of questions for each examinee. Constraint solving in our implementation is based on the Choco constraint solver.¹

In this paper, we show how the problem of multi-exam configuration can be represented as a constraint satisfaction problem (CSP) [14]. We exemplify different types of constraints supported by our configurator and also show

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

✉ ritz@student.tugraz.at (B. Ritz);

alexander.felfernig@ist.tugraz.at (A. Felfernig);

vietman.le@ist.tugraz.at (V. Le); vietman.le@ist.tugraz.at (S. Lubos)

🌐 https://www.tugraz.at/ (B. Ritz); https://www.felfernig.eu/

(A. Felfernig); https://www.tugraz.at/ (V. Le);

https://www.tugraz.at/ (S. Lubos)

>ID 0009-0000-7774-6693 (B. Ritz); 0000-0003-0108-3146

(A. Felfernig); 0000-0001-5778-975X (V. Le); 0000-0002-5024-3786

(S. Lubos)

CC BY 4.0 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://choco-solver.org/>

how the preferences of students can (potentially) be taken into account. In order to analyze constraint solver performance, we evaluate the runtime performance of an open source constraint solver (Choco) on the basis of a typical real-world exam configuration scenario.

The remainder of this paper is organized as follows. In Section 2, we introduce a definition of a multi-configuration task and provide an example from the domain of multi-exam configuration. In this context, we also introduce and exemplify different constraint types. Thereafter, in Section 3, we evaluate the performance of Choco when solving multi-exam configuration tasks also including a performance analysis when solving a real-world configuration task. Threats to validity are discussed in Section 4. The paper is concluded with a discussion of open research issues in Section 5.

2. Working Example

We now give a basic definition of a *multi-configuration task* (see Definition 1) (see [9]) and show how multi-exam configuration tasks can be introduced correspondingly.

Definition 1. A multi-configuration task can be defined as a tuple (V, D, C) with $V = \bigcup\{v_{ij}\}$ is a set of finite domain variables (v_{ij} is variable j of configuration instance i), $D = \bigcup\{\text{dom}(v_{ij})\}$ a set of corresponding domain definitions, and $C = \{c_1, c_2, \dots, c_v\}$ a set of constraints.

In the following, we will use this definition to introduce the task for multi-exam configuration and outline what constraint types are supported by our configurator. In this context, the set of constraints C can be defined by users represented by *instructors* and also *students* (where this is intended).

2.1. Multi-exam configuration

Following Definition 1, a multi-exam configuration task can be defined as follows.

- $V = \{q_{11..q_{nm}}, q_{11..T_1}, q_{11..T_\pi}, \dots, q_{nm..T_1..q_{nm..T_\pi}}\}$ where q_{ij} is question j of the exam of student i , n is the number of exams (students), m is the number of questions per exam, $q_{ij..T_k}$ denotes the value of the k -th *question property* of question q_{ij} , and π represents the number of question properties per question (in our case, $\pi = 6$). Our configurator supports the following question properties:

1. *topic* - topic of the question
2. *level* - difficulty level of the question
3. *min-duration* - minimum estimated time needed to answer the question
4. *max-duration* - maximum estimated time needed to answer the question

- 5. *type* - type of the question (e.g. single/multiple choice, assignment task, etc.)
- 6. *points* - maximum number of points rewarded for correct answers
- $D = \{\text{dom}(q_{11})..dom(q_{nm}), \text{dom}(q_{11..T_1})..dom(q_{11..T_6}), \dots, \text{dom}(q_{nm..T_1})..dom(q_{nm..T_6})\}$ where $\text{dom}(q_{ij}) = \{1..\Omega\}$, with Ω being the total number of questions in the question pool, and $\text{dom}(q_{ij..T_k})$ is a question property domain (one out of the following):
 1. $\text{dom}(\text{topic}) = \{1..\eta\}$ where η is the number of defined question topics
 2. $\text{dom}(\text{level}) = \{1..\mu\}$ where μ is the number of defined question complexity levels
 3. $\text{dom}(\text{min-duration}) = \{1..\tau\}$ with τ indicating the maximum specifiable value
 4. $\text{dom}(\text{max-duration}) = \{1..\kappa\}$ with κ indicating the maximum specifiable value
 5. $\text{dom}(\text{type}) = \{1..\theta\}$ where θ is the number of defined question types
 6. $\text{dom}(\text{points}) = \{1..\phi\}$ where ϕ is the maximum amount of points
- $C = \{c_1..c_v\}$ where c_β is the constraint identifier and v is the number of constraints

Importantly, depending on the question $1..\Omega$ assigned to a question variable q_{ij} , a set of corresponding question properties must hold, for example, if question $q_{11} = 1$, corresponding restrictions such as $q_{11} = 1 \rightarrow q_{11..topic} = A$ indicate the relevant question properties. In Subsection 3.2, we explain in which way we support this aspect in our configuration library. Furthermore, for each student-specific exam i , we need to include an *alldifferent*($q_{i1..q_{im}}$) constraint to avoid situations where a questions is assigned to the same exam twice. In our implementation, this aspect is taken into account on the basis of set variables (see also Subsection 3.2).

2.2. Instructor constraints

Instructor constraints (defined by instructors) in C restrict the set of questions that may or may not appear in exams. Each exam must fulfil all of these constraints. We distinguish between two types of related constraints: *intra-exam* and *inter-exam* constraints.

2.2.1. Intra-exam constraints

Intra-exam constraints restrict which questions are eligible for being part of an exam. Such constraints refer to each individual exam. A simple form of intra-exam constraints is to directly define a specific question property. For example, let us assume that up to now a course has covered only one (the first) topic (A). As a consequence,

the instructor requires that only questions belonging to topic A are part of the first exam (see Formula 1).

$$\forall q_{ij} \in V : q_{ij}.topic = A \quad (1)$$

Furthermore, we might want to restrict the complexity level of questions. For example, assuming that four different question complexities exist, for the final exam the instructor would like to increase the overall exam complexity using an intra-exam constraint specifying that all questions of each exam must have a complexity level of at least 2 (see Formula 2).

$$\forall q_{ij} \in V : q_{ij}.level \geq 2 \quad (2)$$

Intra-exam constraints allow instructors to arbitrarily combine multiple constraints with logical operators. For example, since multiple choice questions can generally be answered rather quickly, an instructor could require that every multiple choice question is of at least difficulty level 3 (see Formula 3 where we assume question type 3 indicates multiple choice questions).

$$\forall q_{ij} \in V : (q_{ij}.type = 3 \implies q_{ij}.level \geq 3) \quad (3)$$

In many cases, instructors would like to be able to specify intra-exam constraints on a more granular level. It is possible to combine intra-exam constraints with a corresponding scope. Constraint scopes enable instructors to specify how many questions per exam need to satisfy a given constraint. To illustrate this aspect, we will continue our previous example (see Formula 1). By the time the next topic (topic B) is covered in the course, the students will have a follow-up exam consisting of 10 questions. The instructor now wants to focus mainly on the new topic. Therefore, they specify constraints such that for each exam only 2 questions belong to topic A and the remaining 8 to topic B (see Formula 4 and 5).

$$\bigwedge_{i=1}^{n(\#exams)} (|\{q_{ij} \in V : q_{ij}.topic = A\}| = 2) \quad (4)$$

$$\bigwedge_{i=1}^{n(\#exams)} (|\{q_{ij} \in V : q_{ij}.topic = B\}| = 8) \quad (5)$$

Constraint scopes also support lower and/or upper bounds, for example, the instructor would like to keep the follow-up exam rather simple. For this reason, between 5 and 10 questions of each exam should be easy to solve, which is indicated by complexity level 1 (see Formula 6).

$$\bigwedge_{i=1}^{n(\#exams)} (5 \leq |\{q_{ij} \in V : q_{ij}.level = 1\}| \leq 10) \quad (6)$$

Instructors may also specify constraint scopes using percentages in order to describe which amount of questions per exam must satisfy the question property constraint. For example, only between 10 and 20 percent of questions per exam should be solvable in less than five minutes (see Formula 7).

$$\bigwedge_{i=1}^{n(\#exams)} \left(0.10 \leq \frac{|\{q_{ij} \in V : q_{ij}.min-duration < 5\}|}{m} \leq 0.20 \right) \quad (7)$$

Intra-exam constraints also support aggregations. In the context of our evaluation settings, we support the functions *sum*, *average*, and *distinct count*. For their application, see the examples in Formulas 8–10.

1. Example (*sum*): The total amount of points per exam is 100 (see Formula 8).

$$\bigwedge_{i=1}^{n(\#exams)} \left(\sum_{j=1}^m q_{ij}.points = 100 \right) \quad (8)$$

2. Example (*average*): The average complexity level of each exam is between 2 and 3. (see Formula 9).

$$\bigwedge_{i=1}^{n(\#exams)} \left(2 \leq \frac{\sum_{j=1}^m q_{ij}.level}{m} \leq 3 \right) \quad (9)$$

3. Example (*distinct count*): Each exam consists of at least 3 different question topics (Formula 10).

$$\bigwedge_{i=1}^{n(\#exams)} (|\{q_{ij}.topic\}| \geq 3) \quad (10)$$

2.2.2. Inter-exam constraints

Similar to intra-exam constraints, inter-exam constraints restrict which questions may be part of exams. However, they constrain how often certain question or question properties may or may not appear in the entire exam configuration. Therefore, inter-exam constraints depend on all exams combined, instead of every exam individually. Such constraints count, for example, how many exams have at least one question that fulfills a given constraint. This sum can be lower and/or upper bounded. For example, we assume that a specific question ξ is part of at least 5 exams but at most 10 (see Formula 11).

$$5 \leq |\{q_{ij} \in V : q_{ij} = \xi\}| \leq 10 \quad (11)$$

As a special case of inter-exam constraints, instructors can restrict the degree of question overlap across exams, i.e., the number of questions that exams have in common. This is especially useful to prevent the generation of identical or very similar exams. The degree of overlap can be lower and upper bounded in order to restrict the

minimum and maximum amount of questions that pairs of exams may share. For example (see Formula 12), the upper bound denotes that no pair of exams exists which shares more than 5 questions, whereas the lower bound states that every pair of exams must share at least 2 questions. This might be useful to create a sense of fairness among students but might lead to cheating. e_λ represents a set of questions comprising all questions of exam λ .

$$\bigwedge_{i=1}^{n(\#exams)} \bigwedge_{j=1}^{n(\#exams)} (2 \leq |e_i \cap e_j| \leq 5) \quad (i \neq j) \quad (12)$$

The amount of exam pairs to be constrained can be further decreased in the context of onsite exams where a pre-defined lecture hall's seating plan (chart) is specified. One goal in such scenarios is to prevent cheating of students positioned in a neighborhood which can be achieved on the basis of constraints avoiding question overlaps in the case of students located next to each other.

Given is the following lecture hall with 5 rows and 8 seats per row (see Figure 1).

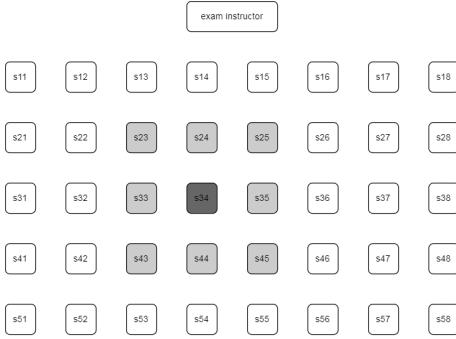


Figure 1: lecture hall seating: seat 34 and its neighbors.

The seats are labeled with the letter s and two digits. The first digit represents its row (top to bottom) and the second digit its position in the row (left to right). We assume no neighboring exams may share even a single question (see Formula 13), where k is the number of neighbors of exam i , e_i is the question set assigned to exam i (student i), and $f(i, j, s)$ describes the set of questions of i 's j -th neighboring exam according to seating chart s .

$$\bigwedge_{i=1}^{n(\#exams)} \bigwedge_{j=1}^k (|e_i \cap f(i, j, s)| = 0) \quad (13)$$

This allows identical exams in the configuration but never for students right next to each other. In order to provide a better understanding of how many constraints approximately need to be added per seat, we have highlighted seat $s34$ (dark gray) and its neighbors (light gray) as an example (see Figure 1). This particular seat has 8 neighbors requiring 8 constraints to be added.

2.3. Student constraints

Student constraints in C can be specified by each student individually. They constrain only the student's exam, no other exams are affected. Student constraints can only further narrow down instructor constraints.

Example: The instructor specifies a constraint such that between 20% and 50% of the questions of each exam must belong to topic A (see Formula 14).

$$\bigwedge_{i=1}^{n(\#exams)} \left(0.20 \leq \frac{|\{q_{ij} \in V : q_{ij}.topic = A\}|}{m} \leq 0.50 \right) \quad (14)$$

Student a decides to restrict this constraint even further so that only a maximum of 25% of questions of their exam belong to topic A (see Formula 15).

$$\frac{|\{q_{aj} \in V : q_{aj}.topic = A\}|}{m} \leq 0.25 \quad (15)$$

3. Evaluation

We now present a performance analysis of our multi-configuration setting.²

3.1. Real world example

We assume that (1) 450 students participate in an exam and (2) a question pool of 45 questions individually associated with one out of four different topic areas (topics) is available. Each exam should consist of $m = 10$ questions. We define the following constraints (C):

1. Each exam should include questions related to at least 2 different topics.

$$\bigwedge_{i=1}^{n(\#exams)} (|\{q_{ij}.topic\}| \geq 2)$$

2. There is at most one multiple choice question.³

$$\bigwedge_{i=1}^{n(\#exams)} (|\{q_{ij} \in V : q_{ij}.type = 3\}| \leq 1)$$

3. 10% – 20% of the questions are assigned to complexity level 4 being the most complex one.

$$\bigwedge_{i=1}^{n(\#exams)} \left(0.10 \leq \frac{|\{q_{ij} \in V : q_{ij}.level = 4\}|}{m} \leq 0.20 \right)$$

4. Each exam should include questions resulting in 40 points in total.

$$\bigwedge_{i=1}^{n(\#exams)} \left(\sum_{j=1}^m q_{ij}.points = 40 \right)$$

²A link to the source code of our configuration approach will be provided in the final paper version.

³Question type 3 was assumed to be multiple choice.

5. Neighboring exams share at most 2 questions (assuming a hall s with 22 rows and 21 seats/row).

$$\bigwedge_{i=1}^{n(\# \text{exams})} \bigwedge_{j=1}^k (|e_i \cap f(i, j, s)| \leq 2)$$

Considering these specific requirements, the configurator yielded a solution in about *one second*.

3.2. Dealing with flexible upper bounds

Instead of relying on a constant number of questions per student exam, it is also possible to support flexible lower and upper bounds. We support this aspect by utilizing Choco *set variables*. Every student exam includes a set of questions. The domain of a set variable is (implicitly) defined by lower and upper bound sets. The lower bound is a set of questions that each exam must include, whereas the upper bound defines the maximum possible question set. In our case, the lower bound is empty and the upper bound equals the question pool.

A varying number of questions per exam could trigger the need for further instructor constraints, for example, to restrict the allowed number of questions. Let us assume a defined question pool with $\Omega = 3$ questions ($\{1, 2, 3\}$) and $n = 2$ exams (e_1 and e_2) represented as Choco set variables (*model* is a Choco *model object*).

```
e1 = model.setVar(lb:{}, ub:{1, 2, 3})
e2 = model.setVar(lb:{}, ub:{1, 2, 3})
```

Now, we want to specify that each exam e_i (of student i) needs to include at least two and at most three questions. In Choco, this constraint would be defined as follows.

```
e1.setCard(model.intVar(2, 3))
e1.setCard(model.intVar(2, 3))
```

In this simplified setting, the possible solution sets for both, e_1 and e_2 are: $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, and $\{1, 2, 3\}$.

If we also want to define restrictions on allowed question properties, the solver needs to know the question properties of each individual question. For scalability reasons, we avoid to define question/property relationships on the basis of constraints. Instead, we support a key-value data structure that allows the identification of question properties on the basis of the corresponding question $ID \in \{1..|\Omega|\}$. Given such a structure, we are now able to define constraints referring to question properties, for example: in each exam, the number of questions of topic A is exactly 2.

In Choco, no related built-in constraints exist. We have defined a custom constraint by extending the *Propagator* class and implemented the two required methods *propagate* and *isEntailed*.⁴ The former is called in each iteration of the solving process. It tries to find solutions by

⁴A GitHub source code reference will be included in the final paper.

counting the number of questions that belong to the specified topic. If the current branch of the solving process cannot satisfy the constraint, a contradiction is indicated. When taking into account this constraint, the possible solutions for e_1 and e_2 are $\{1, 2, 3\}$ and $\{1, 3\}$.

3.3. Evaluation with synthesized data

We have also evaluated the solver performance with synthesized multi-exam configuration tasks along the dimensions of *number of questions* and *number of exams*. Each task utilizes the same 5 constraints as discussed in Subsection 3.1. We choose the lecture hall size depending on the amount of exams n , using the formula $\lceil \sqrt{n} \rceil$, since this is a fairly simple way to assure that all students will fit in the lecture hall and to keep a good ratio between rows and seats per row. The results of this performance evaluation are summarized in Table 1 showing acceptable runtime performances in the context of typical exam settings as well as extreme cases of around 1000 questions and up to 1000 students inducing solver runtimes up to nearly 3 minutes. Notice that a smaller question pool size does not always result in faster runtime.

4. Threats to Validity

In the context of the reported evaluation, we have applied the standard settings of the used constraint solver. A major topic of further work is to further improve runtime performance on the basis of different approaches supporting the learning of solver search heuristics (see, e.g., [15]). Fairness is a crucial aspect to be taken into account when it comes to the automated generation of exams. In this work, we have taken this aspect into account o.a. on the basis exam-specific criteria regarding the percentage of to-be-included questions that are related to a specific complexity level. For future work, we plan to further refine this aspect, for example, on the basis of optimization functions that help to balance the complexity of individual exams on a more fine-grained level. Finally, in real-world settings, we often have to deal with situations where a given set of constraints is inconsistent, i.e., no solution could be identified. In our future work, we will integrate corresponding repair concepts which will help users to find ways out from the so-called *no solution could be found* dilemma. Such approaches can be based o.a. on model-based diagnosis [16].

5. Conclusions

In this paper, we have introduced multi-configuration as a useful approach in scenarios requiring solution set configuration, for example, exam configuration and project

Table 1

Constraint solver (configurator) performance based on synthesized settings differing in terms of number of questions and number of student-specific exams using the constraints introduced in Subsection 3.1. In this context, s =seconds and m =minutes. Cells without unit of measurement represent runtimes in *milliseconds*.

Questions	Exams										
	1	5	10	25	50	75	100	250	500	750	1000
25	14,33	328,33	397,67	349,00	425,33	475,67	516,00	1183,33	3340,33	6,30s	10,10s
50	21,67	34,00	80,33	100,67	143,33	153,00	183,33	448,67	1231,33	2572,33	3769,00
75	26,67	46,33	63,00	88,33	155,33	221,33	234,00	686,67	1689,00	3387,00	5,66s
100	23,67	55,33	80,33	161,00	172,67	318,00	314,00	849,33	2345,00	4899,00	7,36s
150	36,00	93,67	102,67	193,67	301,00	462,00	485,00	1163,33	3751,00	7,08s	12,80s
250	109,00	146,33	185,33	383,33	562,67	725,33	934,33	2540,67	7,21s	14,38s	20,98s
500	114,00	269,33	407,33	878,00	1638,33	2174,33	3020,33	7,64s	19,27s	35,75s	55,30s
750	168,33	415,67	719,67	1677,33	3163,33	5,02s	5,09s	15,92s	33,41s	1,14m	1,50m
1000	250,33	651,67	1052,67	2942,67	5,01s	8,08s	9,04s	27,18s	1,06m	1,81m	2,67m

team configuration. In the context of multi-exam configuration, we have shown a corresponding configuration task representation as a constraint satisfaction problem. We have evaluated the performance of the proposed approach on the basis of an example real-world configuration task as well as a collection of synthesized configuration tasks (differing in terms of the number of pre-defined questions and the number of "to be generated" exams). Our future work will include the integration of further concepts supporting solver performance optimization. Furthermore, we will include features, for example, in terms of optimization functions, that help to take into account aspects such as fairness in a more explicit fashion. Finally, we plan to include concepts that will allow us to take into account historical data, for example, when generating a set of "new" exams, the frequency of questions already "used" in previous exams should be taken into account in order to avoid situations where specific questions are posed too often.

References

- [1] M. Stumptner, An overview of knowledge-based configuration, *AICOM* 10 (1997) 111–125.
- [2] U. Junker, Configuration, in: F. Rossi, P. van Beek, T. Walsh (Eds.), *Handbook of Constraint Programming*, Elsevier, 2006, pp. 837–873.
- [3] L. Hvam, N. Mortensen, J. Riis, *Product Customization*, Springer, 2008.
- [4] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, *Knowledge-based Configuration – From Research to Business Cases*, Elsevier, 2014.
- [5] A. Felfernig, M. Atas, T. Tran, M. Stettinger, Towards group-based configuration, in: *ConfWS'16*, Toulouse, France, 2016, pp. 69–72.
- [6] V. Le, T. Tran, A. Felfernig, Consistency-based integration of multi-stakeholder recommender systems with feature model configuration, in: *26th ACM Intl. Systems and Software Product Line Conference*, ACM, New York, NY, USA, 2022, p. 178–182.
- [7] A. Felfernig, M. Stettinger, G. Ninaus, M. Jeran, S. Reiterer, A. Falkner, G. Leitner, J. Tiihonen, Towards open configuration, in: *ConfWS'14*, Novi Sad, Serbia, 2014, pp. 89–94.
- [8] M. Atas, A. Felfernig, S. Polat-Erdeniz, A. Popescu, T. Tran, M. Uta, Towards psychology-aware preference construction in recommender systems: Overview and research issues, *J. Intell. Inf. Syst.* 57 (2021) 467–489. doi:10.1007/s10844-021-00674-5.
- [9] A. Felfernig, A. Popescu, M. Uta, V. Le, S. Erdeniz, M. Stettinger, M. Atas, T. Tran, Configuring multiple instances with multi-configuration, in: *ConfWS'21*, Vienna, Austria, 2021, pp. 45–47.
- [10] V. Le, T. Tran, M. Stettinger, L. Weißl, A. Felfernig, M. Atas, S. Erdeniz, A. Popescu, Counteracting exam cheating by leveraging configuration and recommendation techniques, in: *ConfWS'21*, Vienna, Austria, 2021, pp. 73–80.
- [11] A. Gotlieb, B. Botella, M. Rueher, Automatic test data generation using constraint solving techniques, in: *ACM SIGSOFT Intl. Symp. on Software Testing and Analysis*, Florida, USA, 1998, pp. 53–62.
- [12] A. Jameson, S. Baldes, T. Kleinbauer, Two methods for enhancing mutual awareness in a group recommender system, in: *Working Conf. on Advanced Visual Interfaces*, Gallipoli, Italy, 2004, pp. 447–449.
- [13] J. Pereira, M. Acher, H. Martin, J. Jézéquel, G. Bötterweck, A. Ventresque, Learning software configuration spaces: A systematic literature review, *Journal of Systems and Software* 182 (2021) 111044.
- [14] F. Rossi, P. van Beek, T. Walsh, *Handbook of Constraint Programming*, Elsevier, 2006.
- [15] M. Uta, A. Felfernig, D. Helic, V. Le, Accuracy- and Consistency-Aware Recommendation of Configurations, in: *SPLC'2022*, ACM, 2022, pp. 79–84.
- [16] R. Reiter, A theory of diagnosis from first principles, *Artificial Intelligence* 32 (1987) 57–95.

Decision Heuristics in a Constraint-based Product Configurator

Matthias Gorenflo¹, Tomáš Balyo¹, Markus Iser^{2,3} and Tobias Ostertag^{1,*}

¹CAS Software AG, CAS-Weg 1 - 5, 76131 Karlsruhe, Germany

²Karlsruhe Institute of Technology (KIT), KIT-Department of Informatics, Karlsruhe, Germany

³University of Helsinki, Department of Computer Science / HIIT, Helsinki, Finland

Abstract

This paper presents an evaluation of decision heuristics of solvers of the Boolean satisfiability problem (SAT) in the context of constraint-based product configuration. In product configuration, variable assignments are searched in real-time, based on interactively formulated user requirements. Operating on user's successive input poses new requirements, such as low-latency interactivity as well as deterministic and minimal implicit product changes. This work presents a performance evaluation of several heuristics from the SAT literature along with new variants that address the special real-time requirements of incremental product configuration. Our results show that the execution time on an industrial benchmark can be significantly improved with our new heuristic.

Keywords

Configuration, Constraint-based Products, Decision Heuristics, Boolean Satisfiability Problem (SAT)

1. Introduction

1.1. Motivation

In wake of an increasing globalization, the demand for customized and personalized products rises in manufacturing and service industries, which previously only utilized the advantages of mass production to offer standardized products for a good value. Shaped by Stanley Davis and his 1987 book *Future Perfect* [1], this new frontier is called *mass customization* and wants to meet the product needs of individual customers. "At its core, is a tremendous increase in variety and customization without a corresponding increase in costs. At its limit, it is the mass production of individually customized goods and services. At its best, it provides strategic advantage and economic value" [2]. This results in increasingly complex models of the product variants that can be configured, if the model shall offer many intertwined parameters a customer is allowed to choose from. Various domains are applicable for product configuration with some of the more complex product models revolving around the assembly of different vehicles. But the decision, if every request of a customer is viable, can become troublesome for even reasonably sized models. In remedying the solving process automatically, so called knowledge-based con-

figuration systems (or simply product configurators) [3] play a key role. The product configurator is a tool to on one hand decide on a product that respects the demands of the user, which can be a customer, while conforming to the limitations and constraints of the manufacturer on the other hand. The description of the product model lies at the core and spans the solution space, which is the set of all possible product variants. This work focuses on an interactive configuration process. During the configuration of a product, a user performs adjustments to the current product by selecting from different parts or properties. Incremental requests coming from the user are called user wishes and the configurator has to check the validity of them.

A method of realizing the constraints of a product model is to utilize propositional logic [4]. Hereby, for a configuration to be valid, it has to satisfy a set of propositional formulas expressing the product configuration problem as shown in [5]. A SAT solver is then capable of checking if all user wishes can be satisfied under the propositional representation of the configurator's specifications. The truth assignment then holds the information about the concrete manifestation of the configured product.

The complexity of the SAT problem is in non-deterministic polynomial time (NP) which can lead to long computation times, exponential in the size of the problem. Contrary to that is the *interactive* nature of product configuration, where users expect a fast response regarding the feasibility of their latest demand. The shorter and faster the time spent on satisfiability checking, the shorter the waiting time for users. Minimizing the time of the SAT solving therefore plays an essential

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

✉ markus.iser@kit.edu (M. Iser); Tobias.Ostertag@cas.de

(T. Ostertag)

✉ 0000-0003-2904-232X (M. Iser); 0000-0003-3294-3807

(T. Ostertag)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

role for interactive, low-latency product configuration.

Luckily, many SAT formulas that model real world problems can be solved quickly thanks to the classic DPLL algorithm [6], which has seen many improvements and additions over the last decades. Sophisticated heuristics play a key role here. This is especially true for decision and branching heuristics, which control the ordering and values of the algorithm's truth assignments [7].

The area of product configuration tends to place different demands on a solver than the classical SAT formulas. The process of incremental product configuration generally leads to less complex computations, since we solve many simple formulas and not just one complex formula.

New challenges arise when the last user requirement cannot be met with a previously selected configuration. The possibility of stating conflicting requirements is intentional, as the user may not be aware of all the interactions between different requirements, or may be in the process of making fundamental changes. In either case, the user relies on feedback from the configurator, and it is the solver's job to calculate a new valid product configuration. This amounts to solving an optimization problem where the new configuration contains as few changes as possible while omitting as few user requirements as possible. To realize this idea, the configurator weights the user's requirements so that changes are associated with costs.

Furthermore, not only a single optimal solution is of interest but every solution with the smallest cost or within a certain delta. So the user is able to select one of the best fitting alternatives. The optimization problem can be realized as a *minimum-cost satisfiability (MinCostSAT) problem* or an equivalent *maximum satisfiability (MaxSAT) problem*. Nevertheless, the additional challenges posed by product configuration also present new opportunities to derive better heuristics for this specific use case.

1.2. Goals and Contributions

This paper lays the focal point on the goal of reaching an as optimal as possible performance of decision heuristics for SAT in the context of incremental product configuration with weighted user requirements as described in the previous section. We implemented four decision heuristics under the additional requirement of deterministic user experience, i.e., the configurator keeps producing the same results. So a user receives the same selection of alternative configurations for every repetition of a specific interactive configuration sequence. The performance of various known branching heuristics from SAT solving as well as new heuristic ideas are evaluated in the product configuration context. We examine how the different sub-formula types and literal types can be exploited effectively in these heuristics.

The implementation of the heuristics and the following

evaluation is done with the product configurator *Merlin CPQ* by the *CAS Software AG*. *Merlin* has a specialized solving process of incremental problems from interactive configuration. Compared to a typical SAT solver, this configurator also supports multiple different formula terms as well as arithmetic expressions.

We evaluate the heuristics with respect to their execution time and the decision count on several product configuration benchmarks and specifically focus our efforts on speeding up the more complex to solve problems to enable fluid and user-friendly configuration of the desired product, even in these taxing cases. Experimental results show that the performance of the especially expensive benchmarks is roughly doubled by the best of the presented branching heuristics.

Interactive product configuration is a unique domain, so general-purpose heuristics do not necessarily achieve the best performance here. In this paper, we introduce new branching heuristics that achieve better performance in the domain of interactive product configuration than the well-known top dogs for more general-purpose benchmarks. It remains to be seen, how the heuristics evaluated in this paper behave in other benchmark domains.

1.3. Related Work

Most modern satisfiability solvers are based on the highly influential foundation of Davis and Putnam [8] and the shortly following DPLL algorithm [9]. Advancements were made regarding decision heuristics, efficient data structures [10], clause learning [11, 12], and search restarts [13, 14]. The effectiveness of this method caught interest in several domains taking advantage of the strong performance of SAT solvers, especially after multiple strong improvements around the turn of the last century. Probably the two largest domains using SAT are automated planning and scheduling [15, 16] and formal verification [17, 18].

Reductions to SAT are also well known in the context of product configuration [3]. Sinz, Kaiser and Küchlin show different methods in [19, 4] that can be deployed for configuration and [5] demonstrates how SAT solvers are able to be used for an interactive configuration process.

The most dominant decision heuristic in SAT solving of the last century is probably Dynamic Largest Individual Sum (DLIS) [20] found in GRASP, the algorithm that revolutionized DPLL and gave birth to the new solving paradigm Conflict-driven Clause Learning (CDCL). The predominant decision heuristic in CDCL solvers in the last two decades has been Variable State Independent Decaying Sum (VSIDS) which was first presented in Chaff [10].

Nevertheless, decision heuristics for CDCL are a vital research area [21]. Application-specific specialized

heuristics are evaluated regularly [22, 23, 24]. Recently, approaches based on reinforcement learning have been successfully used to select heuristics dynamically [25, 26].

Efforts to develop specialized heuristics for product configuration were made in [27] by applying graph analysis to propositional formulas in *Merlin CPQ*. Nevertheless, the resulting heuristic based on coreness is not adaptively reacting to conflicts that will often appear in DPLL and thus could not dethrone VSIDS, which still is *Merlin*'s standard heuristic.

1.4. Overview

Chapter 2 presents the theoretical concepts and definitions covering propositional logic as well as SAT and MaxSAT solving in the context of incremental product configuration. In Chapter 3, we explain the ideas and motivations behind the branching heuristics under consideration. The implementation in *Merlin CPQ* is described in Chapter 4. Subsequently, Chapter 5 presents the results of our performance evaluation of the presented heuristics. Lastly, we provide a summary and our perspective on potential future work in Chapter 6.

2. Theoretical Preliminaries

This chapter is meant to give a brief introduction into necessary preliminaries for our work, but this information is not targeting to be an exhaustive treatise about the field of SAT solving.

2.1. Propositional Logic

In propositional logic, we have two Boolean constants to represent values of "true" and "false". Propositional formulas are built from Boolean variables and operators such as negation, conjunction, and disjunction.

These operators are interpreted with respect to the usual semantics, i.e., the negation of an argument is true if and only if the argument is false, the conjunction of a set of arguments is true if and only if all arguments are true, and the disjunction of a set of arguments is false if and only if all arguments are false.

A variable assignment maps all Boolean variables of a formula to Boolean constants. The truth value of a formula under a given assignment is determined by replacing the variables with Boolean constants accordingly and by successively interpreting the truth value of all sub-formulas according to the operator semantics.

2.1.1. Normal Forms and Clause Types

The most common appearance of SAT formulas is in *conjunctive normal form* (CNF). A CNF formula is a conjunction of clauses. Each clause itself is a disjunction of

one or more literals. A literal is either a Boolean variable (positive literal) or the negation of a variable (negative literal). In contrast to CNF formulas, a formula in *disjunctive normal form* (DNF) if it is a disjunction of terms, where a term is a conjunction of literals.

The *Merlin* product configurator supports formulas constructed as conjunctions of both types of normal forms, CNF and DNF formulas, as well as at-most-one (AMO) constraints over sets of literals. An AMO constraint evaluates to true if and only if at most one of its literals is true. There are several ways to encode this constraint in propositional logic but further details are unimportant for this paper.

2.2. Propositional Satisfiability

The satisfiability (SAT) problem asks the question whether it is possible to find a complete variable assignment that interprets a given formula to true (satisfiable) or declares this impossible (unsatisfiable). A SAT solver can be used to answer such problem instances. We can additionally call a propositional formula valid if it evaluates to true for every possible assignment.

The easiest way to determine satisfiability is achieved by creating a truth table for the whole formula and check whether any resulting value is true. The issue is the effort of this procedure, which grows exponentially with the number of variables.

2.2.1. DPLL Algorithm

DPLL is an enhanced depth-first search algorithm. A partial assignment is successively collected for a given CNF formula by adding literals during its search procedure. The assigned variables are then used to simplify the original formula.

Central to DPLL is unit propagation. If the CNF contains a unit clause (a clauses that consist of only a single literal), then the clause's literal is immediately used to extend the current partial assignment. All further clauses where this literal occurs can then be dropped from the formula because they are satisfied by the assignment. Furthermore, the negation of the literal is removed from all clauses in which it occurs and we call clauses where this happens "touched" (this will be important later). The propagation stops when no more unit clauses are present.

Afterwards, one of three states is reached. If no clause remains, the instance is satisfiable and the algorithm returns a satisfying assignment. If an empty clause emerges, i.e., all literals in that clause are falsified, the instance is unsatisfied under the current partial assignment. This means that decisions have to be undone (backtracking) and if no decision can be undone the instance is unsatisfiable. Otherwise, we need to heuristically pick a decision variable which we use to extend the current

partial assignment. Two recursive calls have to be made now; the variable is assigned true in one branch and false in the other.

This algorithm covers all possible branches in the worst case, which makes it sound and complete but also not better than the naive approach regarding this aspect. However, the performance on real world problems is commonly clearly superior to the worst-case performance due to unit propagation and further techniques.

2.2.2. Clause Learning

A major improvement made to DPLL-based solvers has been the concept of conflict-driven clause learning (CDCL), first shown in the GRASP solver [11, 12] and then advanced by Chaff [10]. It has been shown that CDCL *p*-simulates general resolution which makes CDCL strictly more powerful than classic DPLL [28]. In CDCL, each time a decision leads to an empty clause, the conflicting assignment is analyzed to derive a new clause that is added to the formula. The idea behind that is to avoid repeating mistakes across different similar branches in the search. Additionally, conflict analysis can determine a backtracking level for directly backtracking multiple decision levels instead of one at a time. A slightly closer look at this conflict analysis will be taken in the next chapter, where we discuss heuristics that operate on the implication graph that is used for conducting such a conflict analysis.

2.3. Propositional Optimization

The common formalism for describing propositional optimization problems is the weighted Maximum Satisfiability Problem [29]. A MaxSAT instance consists of hard clauses and soft clauses and associates a weight with each soft clause. A solution for a MaxSAT instance is characterized such that it satisfies all hard clauses and the sum of weights of satisfied soft clauses is maximized. In contrast, Minimum-Cost Satisfiability (MinCostSAT) associates weights with every variable to define the cost of an assignment. In that case, the target is to find a solution of lowest possible cost. The reduction of MinCostSAT to MaxSAT is straight-forward through adding the variables as negative literals in a soft unit clause with a weight according to the variable's cost.

Merlin CPQ uses such costs on the product's properties in the previous step (since we want to avoid overriding preconfigured parts as much as possible) as well as for the user wishes if they cannot all be realized together. In order to achieve this goal, *Merlin's* DPLL implementation supports best-first search, that prefers branches of minimal cost. This search follows the branches that cost the least. Should a branch appear that increases the cost above the currently optimal path's cost, then the

search is interrupted and always greedily continued at another branch that sits on the path of minimal cost. A relaxed version with a reduced memory consumption is the beam search. Instead of potentially exploring all promising branches, only a maximum number of the cost-optimized children (defined by the beam width) are considered.

3. Decision Heuristics

Solving SAT instances under the given optimization goals is computationally hard. Nonetheless, DPLL in combination with good heuristics can often efficiently solve many formulas that model real world problems. Thus we are taking a look at several decision heuristics from SAT solving. In this chapter, we review and present well-known and new heuristics that implemented and evaluated in the context of product configuration in *Merlin CPQ*.

Important regarding possible heuristics for *Merlin* is also that they have to pick their decision from a pre-defined set of candidate literals. These candidates stem from the unit propagation's touched clauses. The touched clauses unassigned literals are candidate literals if the clauses are still unsatisfied when the heuristic is called.

3.1. Variable State Independent Decaying Sum

Variable State Independent Decaying Sum (VSIDS) is one of the most efficient decision heuristic for SAT solving since over 20 years. The original idea arose for the solver Chaff [10]. VSIDS was then refined and adjusted slightly over the years, for example in MiniSat [30]. The common idea behind the heuristic is that every variable maintains a counter. The unassigned variable with the highest counter value is chosen for the branching when the solving procedure requires a next decision. The counters are maintained as follows:

- Initially, every counter starts with a score that is typically set to zero. Alternatively, the initial score could also be the amount of occurrences of the respective variable in the propositional formula.
- The counter is incremented each time the respective variable is involved in the reasons for a conflicting assignment. With Chaff the focus was only on the learned clauses, MiniSat expanded the involvement to all clauses that appeared during conflict analysis.
- All counters are periodically decreased by a constant factor after a certain amount of conflicts occurred. This so called decaying is meant to give higher priority to variables that appeared in more recent conflicts.

3.2. Distance Heuristic

The distance heuristic pursues a promising approach as described in [31]. This branching heuristic is based on counters like VSIDS and also makes use of decaying. However, the score increment dynamically takes into account an estimate of how much a variables contributes to the conflicting assignment under analysis. This is done by taking into account the position of the variables in the implication graph that is commonly used in CDCL for analyzing the reasons for a conflicting assignment.

An implication graph is a directed acyclic graph. Each node in the implication graph either represents a clause that triggered an assignment during unit propagation (including the empty clause which triggered the conflict under analysis) or a decision literal. In the implication graph, edges represent propagated, i.e. implied, literal assignments. Each edge is rooted in the node representing the clause or decision that is the reason for the assignment and ends in the node representing the clause in which this assignment falsifies a literal. The creation process starts by representing the conflicting clause with a node and subsequently adding incoming edges for each falsified literal, rooting them in nodes representing the respective reasons for their assignment.

The authors hypothesize that the fewer clauses a variable depends on during the conflict, the higher the probability that the variable contributes to a later conflict. Unfortunately, realizing this dependence hypothesis exactly would be too computationally intensive. The distance heuristic is an approximation to this idea, in that it determines the number of vertices that are located on the longest path in the implication graph from the node representing the reason of a variable's assignment to the node representing the conflicting clause.

The distance heuristic scores literals differently depending on their responsibility for the conflict in contrast to the constant increment of VSIDS. This heuristic aims to be more precise because of the more elaborate scoring, especially during the beginning of the search where only few conflicts occurred and none of the decision heuristics is sufficiently initialized yet. However, the authors of the distance heuristic point to the computational overhead of their heuristic as compared to VSIDS. In their empirical evaluation, they found that it is better to switch from the distance heuristic to VSIDS after 50,000 conflicts. This seems especially fitting for product configuration because the focus on interactivity means that the targeted problems are typically faster to solve compared to some large SAT instances running several minutes and resolving tens of thousands of conflicts.

3.3. Conflict Heuristic

While the previously described heuristics increase the scores of variables that appear in reason clauses or learned clauses, we thought that in our context we could try something that is much simpler. The conflict heuristic is our own variant of VSIDS, where we simply increase the scores of variables in the conflicting clause. The intuition behind this procedure is that such variable assignments should be fixed as quickly as possible, ideally to satisfy as many unsatisfied clauses as possible.

3.4. Heuristics based on Pure Literals

Pure literals were already used in the original algorithm of Davis and Putnam in the affirmative-negative rule [8]. After each unit propagation, the rule searches for literals whose negated form does not occur in the formula under the current partial assignment. Such literals can be set to true without conflict, so that clauses containing them can be eliminated from the formula. This procedure was later called "pure literal elimination". However, this has disappeared from most modern CDCL solvers because the computational overhead involved is usually not compensated by the benefits of this instance simplification procedure. This is due to the advent of efficient data structures for unit propagation which are commonly used in modern SAT solvers.

Merlin uses different data structures for unit propagation, primarily due to the additional requirements imposed by incremental user interaction, which requires the ability to correct assignments deterministically and non-chronologically. But the non-chronological correction of assignments is also detrimental to the applicability of the pure literal rule. At any point in time, a user could select a property such that a previously pure literal is no longer pure. This would not be possible in classical SAT solving, where a pure literal appearing under a particular variable assignment remains a pure literal at least for all branches below it. However, assigning a pure literal explicitly by a branching decision is a valid option.

3.4.1. Pure Literal Heuristic

A first idea for a decision heuristic that utilizes this concept can thus try to select pure literals whenever possible and thus add them to the partial truth assignment. If there are no pure literals available, then the remaining variables are handled afterwards with a different strategy. In the basic variant this is done by a fixed initial ordering.

3.4.2. Pure Literal Phase

The approach of favoring pure literals can and should though be combined with other heuristics as well, where those heuristics proceed instead of the random ordering.

Additionally, preferring pure literals can be done in certain phases. So pure literals could only be preferred after a certain amount of conflicts or decisions, up to a certain amount, within a range, or by having alternating phases of favoring pure literals and just the basic heuristic.

3.4.3. State Dependent Pure Literals

We try some modifications related to keeping pure literals up-to-date during run time for potential improvement. A heuristic can remember so-called "almost" pure literals, which we see as variables that are mostly present in one polarity, while the other polarity is rare. For every almost pure literal, the configurator also stores the clauses preventing the literal from being completely pure. If all the clauses related to an almost pure literal are marked deleted during unit propagation, then this literal is being added to the set of pure literals. This would thus increase the amount of available pure literals.

3.4.4. Pureness Variant

Pureness of literals is another variant of pure literals. The amount of positive versus negative literals of each variable determines a pureness percentage. Variables with a balanced share of positive and negative appearances lead to a low pureness, almost pure literals get a high pureness percentage, and fully pure literals would be 100 percent. The decision heuristic then picks the literal with highest pureness score. Again, different heuristics can be used as combination, either as tie breaker or to set multiple scores off against each other.

3.5. Clause-Based Heuristic

For the clause-based heuristic (CBH), all clauses are kept in an ordered list and decision literals are picked from the top-most unsatisfied clauses in that list according to [32]. To create an initial ordering of clauses, the priority of a clause is calculated based on the number of occurrences of the literals it contains. To increase the proximity of clauses that have literals in common, the ordered list is gradually populated by starting with the clause with the highest priority and then first increasing the priorities of all clauses that have literals in common with the clause just added. This process is repeated until all clauses are added to the list. The ordered list is then updated on each conflict by moving the conflicting clause and the reason clauses to the front. For the decision heuristic to select a literal from the top-most unsatisfied clauses, each literal has additional counters to measure its contribution to (recent) conflicts, while larger scores are preferred. Further details can be found in [32].

4. Implementation

Merlin CPQ already contains implementations of VSIDS and a few other heuristics. In this section we discuss implementation details of VSIDS, CBH and CBH Simplified.

4.1. Variable State Independent Decaying Sum

VSIDS is already part of *Merlin* and the default heuristic. The concrete realization of VSIDS is quite different from the ones used in off-the-shelf SAT solvers. One difference is that Merlin initially selects variable's based on their individual occurrence counts in the formula. This is in contrast to most implementations which initialize scores with zero (cf. CaDiCaL [33], or MiniSat [30]) The variable occurrence counts and the VSIDS score of a variable are kept separately. The sum of both scores is then used to determine the final score used with decisions.

During conflict analysis, the VSIDS scores are incremented by a bump value which is initially set to 100. The purpose of VSIDS decay is to make previously bumped variables less important over time. In *Merlin*, this is realized by increasing the bump value over time. Every 15 conflicts the bump value is increased by two percent.

4.2. Clause-Based Heuristic

An exact reimplementation of the clause-based heuristic as it is described in [32] would not suit *Merlin*'s heuristics, who have to make their decision based on the given set of candidate literals. Therefore, we do not explicitly manage a global clause-list as it is done in the original CBH. Only initially, a list based on all clauses is created with the ordering described by the CBH authors. Hereby, we go through the original set of all clauses and remember for each unique literal, in which clauses it was contained. Then we create a new list with every variable, which is sorted regarding the variables' score based on the counters of the variable's two literals. The paper would now put the clauses containing the highest scoring variable into the clause list, however, we assign a priority to the literals depending on their supposed position in this list instead of explicitly storing the clause list for later usage. So in the first step, the highest scoring variable and all variables that share a clause with it receive priority -1. Each variable has two additional local literal counters which get increased for every occurrence of its respective literal as part of a clause added to the clause list. Therefore, we also increase this local score when setting the priority. The priority is afterwards decreased by one for the next bunch of clauses. This procedure now continues until every variable was selected, while the local score is always added to the variable's score.

Updates due to a conflict can then be realized by calculating the conflict responsible clauses. Each literal of these clauses receives a new priority (starting at zero) that is one higher than the previous highest priority. This procedure is equal to moving clauses to the topmost position of a clause list. Additionally, these literals get a bump to their conflict counter. The same then happens to the actual conflict clause with the next priority value.

Choosing a literal can then be done by checking the position of a literal via its priority. The ones with the highest priority (which is equal to the topmost position in a clause list) are preferred. Literals from the supposed topmost clause will receive the same priority, thus the best literal is further selected depending on a variable's counters that get bumped for conflicts. The official clause-based heuristic seems a bit over-engineered because it keeps counters that get only increased during a conflict and ones that additionally decay periodically. This decay seems of questionable importance since we already move literals that occur in recent conflicts to the top. Thus we do drop this second counter in our implementation.

CBH also operates rather complex in regard to the different counters for the variables because for every comparison of two variables, several counters need to be added and multiplied together for each of the variables. This may cause unnecessary overhead. The formulas that are used to combine the counters of a positive and negative literal into a variable's counter are the main reason behind CBH's need for this dynamic calculation. In all these formulas, the positive and negative literal get added to three times the minimum of the two literals. The reason behind this last term is to punish variables where just one of its literals is important, kind of like the inverse of pureness. But the CBH authors do not specify how strong its impact on the heuristics performance is. So we create a simplified clause-based heuristic regarding the scoring functions by removing the minimum component. Consequently, we also drop the distinction between positive and negative literals in regard to the score and directly add them up. This means we explicitly combine values from a positive and negative literal pair into one variable count, since the distinction will be unimportant for our simplified heuristic. As a welcome side effect, this combination also reduces the number of individual counters that need to be stored.

5. Evaluation

The evaluation of the heuristics is done on several problems from three rule sets in *Merlin CPQ*. These rule sets are based on the use cases of customers who use *Merlin* for product configuration. Our testset contains a total of 241 individual benchmarks across 85 methods of customizing trucks. Each test translates to one user wish

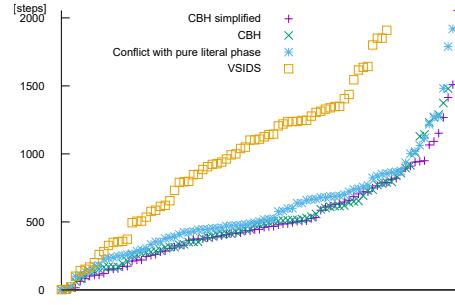


Figure 1: Evaluation with respect to decision count.

and a method with multiple tests translates to a series of user requests. The most important statistics of the formulas used in testing are as follows: 28000 variables, 127000 clauses, 5000 other constraints.

The performance metrics we consider for the problems in this evaluation are the execution time, averaged over three runs, and the sum of the amount of contradictions that were encountered plus the number of branches that were taken. The number of branches taken due to decisions is a good indicator of the heuristic's effectiveness. Contradictions happen less often, ideally a few hundreds for the most elaborate methods and at most around five to ten percent the amount of branches. Therefore we included it as a small contribution on defining the effectiveness of the decisions selected. Both numbers should be as low as possible and the sum of them is called "steps" in the following for simplicity. Their advantage is that the amount of contradictions and branches is deterministic in every run and hardware agnostic. The only downside is that potential time-consuming calculations of a heuristic remain hidden when just looking at the step count. Therefore we also measure the concrete execution time (in milliseconds) for certain comparisons where we expect overheads from parts of a heuristic.

Every metric is always plotted per test method. Cactus plots – typically seen at SAT and SMT competitions – are used whenever more than two heuristics are compared. Here, the methods are ordered by importance for each heuristic element displayed in the graph. The top and right borders represent the timeout limit.

All measurements are performed on an Intel Core i7-4710MQ CPU at 2.5 GHz with 16 GB of RAM under Windows 10, version 21H1. The *Merlin CPQ* version in use is the state of the master branch on 9. December 2021 and running JDK 11.0.13.8.

In Figure 1 we can see that the quality of the literals chosen by CBH are rather good, meaning that it takes just a few decisions to finish the algorithm for most tests

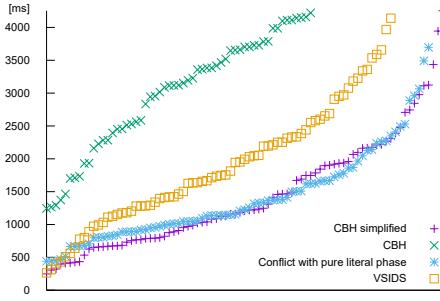


Figure 2: Evaluation with respect to execution time.

– typically even less than the pure literal modified VSIDS variants. However, looking at Figure 2 we see that the complicated summation used by CBH comes at a huge penalty for the execution time. The modified and simplified alternative eliminates this issue.

The CBH simplified heuristic works as intended. It keeps the strength of CBH’s good decision making while maintaining to reach this conclusion quickly and without too much overhead. The initial priority should also be calculated more efficiently. So the execution time across the whole of rule set 1 is therefore always competitive and often even faster than the best VSIDS variants.

6. Conclusion

We presented and implemented several heuristics and tweaks that were able to improve the decision making of *Merlin CPQ*. The largest tests from an industrially used rule set could be significantly improved by an optimized version of the clause-based heuristic. Our idea of a branching heuristic that specializes on the variables causing conflicts and utilized pure literals as preference also performed almost as strong.

In a configurator, the initial weighting of variables is substantial for the performance. The main conclusion for a product configuration heuristic is the significance of selecting interrelated variables. There are several ways a configurator can group them together and take advantage of them. Keeping them united according to their occurrence in clauses is what worked best for our benchmarks. Other groupings are possible according to concepts like pure literals, the types of literals and their clause type they are contained in, or their appearance in the start configuration. The aim is to offer a few good heuristics to the people who design the rule sets that are then brought to users who can configure their product. The designers typically test several heuristics with their specific

environment and are then able to choose the one that performs best for them.

There are still many promising heuristics that can be tested in the field of product configuration, as well as countless combinations and alternations. The currently popular Learning Rate Branching (LRB) [34] based on reinforcement learning is a prime candidate. However, it is important to consider the limitations of the product configurator’s structure. Some information might not be directly accessible to the decision heuristic and only realizable with major changes to the whole architecture. Ideas that contain parts which are inefficient to calculate might need to be changed to perform as desired. So a heuristic has to be adopted to the system it is used in, otherwise the configuration system would be in certain aspects designed around the decision heuristic.

References

- [1] S. Davis, Future Perfect, Basic Books, 1987.
- [2] B. Pine, J. Pine, S. Davis, H. B. Press, Mass Customization: The New Frontier in Business Competition, Harvard Business School Press, 1993.
- [3] D. Sabin, R. Weigel, Product configuration frameworks - a survey, IEEE Intell. Syst. 13 (1998) 42–49.
- [4] C. Sinz, A. Kaiser, W. Küchlin, Formal methods for the validation of automotive product configuration data, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 17 (2003) 75 – 97.
- [5] M. Janota, Do sat solvers make good configurators?, in: SPLC, 2008.
- [6] M. D. Davis, G. Logemann, D. W. Loveland, A machine program for theorem-proving, Commun. ACM 5 (1962) 394–397.
- [7] H. Katebi, K. A. Sakallah, J. Marques-Silva, Empirical study of the anatomy of modern sat solvers, in: SAT, 2011.
- [8] M. D. Davis, H. Putnam, A computing procedure for quantification theory, J. ACM 7 (1960) 201–215.
- [9] D. W. Loveland, A. Sabharwal, B. Selman, DPLL: The core of modern satisfiability solvers, in: Martin Davis on Computability, Computational Logic, and Mathematical Foundations, 2016.
- [10] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, S. Malik, Chaff: engineering an efficient sat solver, Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232) (2001) 530–535.
- [11] J. Marques-Silva, K. A. Sakallah, Grasp-a new search algorithm for satisfiability, Proceedings of International Conference on Computer Aided Design (1996) 220–227.
- [12] J. Marques-Silva, K. A. Sakallah, Grasp: A search algorithm for propositional satisfiability, IEEE Trans. Computers 48 (1999) 506–521.

- [13] G. Audemard, L. Simon, Refining restarts strategies for sat and unsat, in: CP, 2012.
- [14] A. Biere, A. Fröhlich, Evaluating cdcl restart schemes, in: POS@SAT, 2018.
- [15] H. A. Kautz, B. Selman, Planning as satisfiability, in: ECAI, 1992.
- [16] H. A. Kautz, B. Selman, Pushing the envelope: Planning, propositional logic and stochastic search, in: AAAI/IAAI, Vol. 2, 1996.
- [17] A. Biere, A. Cimatti, E. M. Clarke, Y. Zhu, Symbolic model checking without bdds, in: TACAS, 1999.
- [18] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, Y. Zhu, Bounded model checking, Adv. Comput. 58 (2003) 117–148.
- [19] C. Sinz, A. Kaiser, W. Küchlin, Detection of inconsistencies in complex product configuration data using extended propositional sat-checking, in: FLAIRS Conference, 2001.
- [20] J. Marques-Silva, The impact of branching heuristics in propositional satisfiability algorithms, in: EPIA, 1999.
- [21] A. Biere, A. Fröhlich, Evaluating cdcl variable scoring schemes, in: SAT, 2015.
- [22] J. Rintanen, Planning as satisfiability: Heuristics, Artif. Intell. 193 (2012) 45–86.
- [23] M. Iser, M. Taghdiri, C. Sinz, Optimizing minisat variable orderings for the relational model finder kodkod - (poster presentation), in: A. Cimatti, R. Sebastiani (Eds.), Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings, volume 7317 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 483–484.
- [24] P. Beame, H. A. Kautz, A. Sabharwal, Towards understanding and harnessing the potential of clause learning, CoRR abs/1107.0044 (2011).
- [25] M. S. Cherif, D. Habet, C. Terrioux, Combining VSIDS and CHB using restarts in SAT, in: L. D. Michel (Ed.), 27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021, volume 210 of *LIPics*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 20:1–20:19.
- [26] D. Speck, A. Biedenkapp, F. Hutter, R. Mattmüller, M. Lindauer, Learning heuristic selection with dynamic algorithm configuration, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 31, 2021, pp. 597–605.
- [27] S. Haug, Graphentheoretische optimierung der satberechnung im anwendungsfall produktkonfiguration, 2021.
- [28] K. Pipatsrisawat, A. Darwiche, On the power of clause-learning SAT solvers as resolution engines, Artif. Intell. 175 (2011) 512–525.
- [29] C. M. Li, F. Manyà, Maxsat, hard and soft constraints, in: Handbook of Satisfiability, 2021.
- [30] N. Sörensson, N. Eén, Minisat v1.13 - a sat solver with conflict-clause minimization, 2005.
- [31] F. Xiao, C. Li, M. Luo, F. Manyà, Z. Lü, Y. Li, A branching heuristic for sat solvers based on complete implication graphs, Science China Information Sciences 62 (2017) 1–13.
- [32] N. Dershowitz, Z. Hanna, A. Nadel, A clause-based heuristic for sat solvers, in: SAT, 2005.
- [33] A. Biere, K. Fazekas, M. Fleury, M. Heisinger, CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020, in: T. Balyo, N. Froyles, M. Heule, M. Iser, M. Järvisalo, M. Suda (Eds.), Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions, volume B-2020-1 of *Department of Computer Science Report Series B*, University of Helsinki, 2020, pp. 51–53.
- [34] J. H. Liang, V. Ganesh, P. Poupart, K. Czarnecki, Learning rate based branching heuristic for sat solvers, in: SAT, 2016.

Identifying Potential Applications of Service Configuration Systems in a Logistics Company

Erika M. Strøm¹, Tine M. Münsberg¹ and Lars Hvam¹

¹ Technical University of Denmark, Koops Allé 404, 2800 Kgs. Lyngby, Denmark

Abstract

The logistics industry is challenged by the demand for customized services. This challenge is especially apparent to third-party logistics (3PL) companies offering warehousing services. These services include inbound, storage, and outbound logistics along with value-added services (VAS) based on client requirements. 3PL companies must deliver customized services to a wide range of clients while remaining efficient and competitive. Product configuration systems (PCS) have shown to play a key role in providing customized products efficiently. However, literature on the application of configuration systems to services is limited. This paper investigates the potential applications of a service configuration system (SCS) in a logistics company offering customized warehousing services. The aim of this paper is to identify potential applications of a configuration system to a company offering a large variety of services. Modularization of services and applications of configuration systems are shortly investigated and a framework to identify possible applications of PCS is adjusted and applied to services. The framework is applied to a 3PL company to identify potential applications of SCS and to assess the difference between application of configuration systems to products and services. The study uses qualitative data collected in interviews from the case company. The identified applications provide a basis for further scoping of configuration projects in the logistics company.

Keywords

Configuration Systems, Services, Logistics Companies, Third-Party Logistics (3PL), Applications

1. Introduction

Third-party logistics (3PL) companies operate in a highly dynamic and competitive environment [1–3]. These companies manage logistics operations for a wide range of clients from different industries [2]. The warehousing segment manages multi-client warehouses with concurrent logistics operations for different clients [1, 4]. These operations include inbound, storage, and outbound logistics along with value-added services (VAS) [1, 5]. VAS are activities that go beyond the basic logistics operations and add value to the product. VAS are customized to the individual client and encompass a wide range of services, such as labelling [3] or returns handling. Logistics companies can achieve higher profits from providing VAS as the profit margin on simply moving and storing goods is minimal. VAS also enable logistics companies to build closer relationships with clients and differentiate themselves from the competition [3]. The ability to provide VAS plays an important role in gaining a competitive advantage. However, the high level of customization generates several challenges such as complex and time-consuming specification processes for new clients.

The challenges associated with customized services are similar to those of customized products. The concept of mass customization describes the delivery of customized products using mass production techniques to achieve similar lead time, prices, and quality to mass produced products [6]. Product configuration systems (PCS) have shown to be key drivers when enabling mass customization of products. PCS support the development of specifications for customized products [6, 7]. This process includes the activities related to gathering client information, configuring products, and generating product specifications [6]. Case studies have shown the implementation of PCS to reduce man-hours and lead-time related to preparing quotations along with reduced errors in the specification process [6–9].

The development and implementation of configuration systems for products is described in numerous literature [6, 7, 9]. However, limited literature describes configuration systems for services. This poses the question of whether configuration systems can be applied to logistics services to yield similar benefits as with products. The logistics services described in this paper have an undefined solution space as the services can be configured in almost infinite combinations to meet the individual clients' needs. Parallels can be drawn to engineer-to-order (ETO) companies with similar

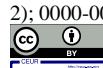
ConfWS'23: 25th International Workshop on Configuration, Sep 06–07, 2023, Málaga, Spain

EMAIL: emst@dtu.dk (A. 1); time@dtu.dk (A. 2); lahv@dtu.dk

(A. 3)

ORCID: 0000-0002-7123-2832 (A. 1); 0000-0001-8924-8026 (A.

2); 0000-0002-7617-2971 (A. 3)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

undefined solution space and complex processes [6, 10]. Due to this complexity, multiple configurators or PCS are often required to support the specification process in ETO companies and the implementation process is gradual [11]. Kristjansdottir et al. [11] describe a need for identifying and prioritizing PCS projects before initiating development of PCS in ETO companies. This may also apply to logistics services.

This paper aims to contribute to literature on configuration systems for services and provide a framework for identifying potential applications of service configuration systems (SCS). The framework is based on a framework for ETO companies developed by Kristjansdottir et al. [11] and adjusted to services based on the studied literature. The framework is applied to a case company offering a wide range of logistics services.

The paper is structured as follows: Section 2 describes the literature within services and configuration systems and Section 3 explains the research method used in this study. Section 4 describes the adjusted framework for services and Section 5 describes a case study of a 3PL company. The framework and case study are discussed and concluded upon in Section 6 along with considerations for future research.

2. Literature review

The literature review describes existing literature within modularization of services and application of configuration systems. Previous application of configuration systems to services is described and the research gap is identified.

2.1. Modularization of services

Modularization and the use of modules is closely related to product configuration [6, 12]. Modularity and product platforms enable companies to efficiently develop and produce products to meet the needs of different market segments [13]. Configuration systems build products from modules according to a set of defined rules and constraints [6]. Hvam et al. [6] describe one of the problems associated with developing and implementing configuration systems as a lack of clearly defined product families and a lack of consensus regarding which variants to offer or which market segments to serve.

Research in the field of service modularity is limited [14–17]. It is an apparent challenge for service companies to define standard variants due to the nature of services [16]. Christensen [18] defines services as intangible, heterogenous, and perishable. Services are a co-creation between the clients and the service industry and are simultaneously produced and consumed [18]. Chervonnaya [19] describes customer behavior in services as volatile and unpredictable. Several authors argue that the concept of modularity could benefit service-based companies [15, 16, 20]. Løkkegaard et al. [16] argue that service companies should adopt the concept of modularization to remain competitive in a growing and dynamic sector.

Some authors have defined modularization in the context of services. Pekkarinen and Ulkuniemi [14] describe a modular service consisting of several service

modules. A service module is “one or several service elements offering one service characteristic” [14] and a service element is the smallest unit a service can be divided into [14]. Tuunanen et al. [21] adopt a similar definition of modular services and service modules. Pekkarinen and Ulkuniemi [14] define three dimensions of modularity with interfaces between each dimension: (1) modularity in services, (2) modularity in processes, and modularity in organization. Generally, there are different interpretations of service interfaces [15, 16].

Pekkarinen and Ulkuniemi [14] and Løkkegaard et al. [16] develop a conceptual model for service platforms. The authors incorporate market segmentation in both models as applied to products in the Power Tower by Meyer and Lehnerd [13]. The authors argue that it is important to understand the market in a platform context due to the service industry’s dynamic environment and the heterogenous nature of services.

2.2. Application of configuration systems

Configurators are widely developed and used in the manufacturing industry [7, 9, 11]. Numerous articles describe strategies and approaches for developing configuration systems for manufacturing companies [7, 9]. Kristjandottir et al. [11] argue that the existing strategies do not provide guidelines for identifying different applications of PCS. Kristjansdottir et al. [11] develop a framework to identify possible applications of PCS in ETO companies. The study aims to address the challenges in identifying and prioritizing PCS projects prior to the development process. The authors argue that this step is particularly important in ETO companies where multiple PCS support the specification processes. Thus, there is a need for a structured approach to breakdown and prioritize PCS projects. The framework is divided into three overall steps: The first step identifies potential PCS, the second step aligns the PCS with IT systems, and the third step provides an overview of the possible applications of PCS [11].

Literature describing approaches for applying configuration systems to services is limited. Christensen [18] applies a procedure for products by Hvam et al. [6] to services and suggests an adapted approach to mass customize service level agreements based on a service variant master to map the service platform. Some of the identified issues are defining the different types of services which is important for the quality of the service delivery and modularizing services with existing theories for products [18]. Mueller et al. [9] suggest a five-step approach for developing and implementing configurators for commissioning services in ETO companies. The application to a case study showed that ETO companies can achieve a significant reduction in specification time and resources for commissioning services. The authors develop the approach based on strategies for PCS. The first step aims to scope the commissioning configurator project. The authors argue that it may be beneficial to only include part of the commissioning service, but guidelines on how to scope this are not described [9].

The literature does not describe how to identify potential applications of configuration systems for services. Thus, there is a need for a structured approach to identify potential applications of SCS.

3. Research method

The research method is divided into two sections: Development of the framework and validation of the framework. The first section describes the development of a framework to identify potential applications of configuration systems to services based on the framework for PCS. The second section explains the process of validating the framework in a logistics company.

3.1. Development of the framework

The development of the framework is based on the literature within services and configuration systems and knowledge about the case company. The literature contributed to an understanding of (1) existing research within modularization of services along with relevant definitions and methods and (2) the importance of identifying potential applications of PCS to complex products and processes and existing case studies on the development of configuration systems for services. The framework was based on the framework for PCS by Kristjansdottir et al. [11] and adjusted to services based on literature and discussions within the research team.

3.2. Validation of the framework

The framework is validated through a case study in a logistics company offering warehousing services. A case study was chosen as it allows the study of phenomena in its natural setting and generation of theory from real-world observations [22].

The case company, also referred to as *the company*, selected for the study has very limited experience with configurators and no experience with large configuration projects. Multiple stakeholders in the company have expressed an interest in the application of configuration systems and potential areas of application have been proposed in previous projects. However, a structured approach for identifying potential applications of configuration systems in the company is lacking.

The framework was applied to the case company by the researchers for validation. The data used in the case study was collected from semi-structured interviews with different stakeholders in the case company. The framework and the generated results were presented to managers from different business units and the feedback was used to improve the framework for services.

4. Framework for identifying potential applications of SCS

The proposed framework to identify applications of SCS consists of three overall steps derived from Kristjansdottir et al. [11]. The steps are as follows: (1) Identifying potential SCS, (2) Aligning IT development, and (3) Establishing an overview of SCS applications. Figure 1 shows the framework. The individual steps and sub-steps are described in the following sections.

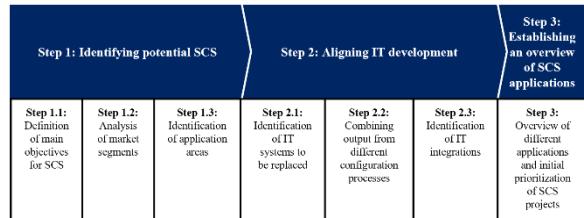


Figure 1: The proposed framework to identify applications of SCS.

4.1. Step 1: Identifying potential SCS

The aim of step 1 is to identify potential SCS. The step is divided in three sub-steps: Step 1.1 defines the main objectives for SCS, step 1.2 analyzes market segment, and step 1.3 identifies application areas.

4.1.1. Step 1.1: Definition of main objectives for SCS

The aim of this step is to identify the main objectives for the SCS. Kristjansdottir et al. [11] describe how objectives are important as they influence decision-making when evaluating application areas in step 1.3 and when evaluating the overview of the SCS applications in step 3. This step also applies in the process of identifying applications for SCS. The intangible and heterogeneous nature of services may emphasize objectives surrounding standardization of specification processes, e.g. guidelines or pricing strategies to ensure specification processes are consistent and efficient. Likewise, there may be an enhanced need for uncovering and understanding client needs as services can be modified during operation which requires adjustments to specifications along with additional resources and potentially higher costs.

4.1.2. Step 1.2: Analysis of market segments

The aim of this step is to analyze the market segments served by the company. This step is added to the framework because the company must assess which market segments to serve and which services to include in the configuration system. As described by Hvam et al. [6], one of the problems associated with developing and implementing a configuration system is the lack of clarification and consensus regarding the variants to include. Market segmentation is used to define and differentiate service variants as described by Løkkegaard et al. [16].

4.1.3. Step 1.3: Identification of application areas

In step 1.3 potential application areas of the configuration system are identified. These application areas can be split into a commercial and a technical configuration process as described by Forza and Salvador [23]. The product is described with the specifications determined by the customer in the commercial

configuration process, while each distinct product variant is described in the technical configuration process, e.g. with bill of materials (BOM). [23] The same could apply to services.

The objectives identified in step 1.1 serve as guidelines for application areas along with the list of questions described by Kristjansdottir et al. [11].

4.2. Step 2: Aligning IT development

The aim of this step is to understand the current IT systems supporting the specification processes and to align with the configuration system [11]. The second step is divided into three sub-steps: Step 2.1 identifies IT systems to be replaced, step 2.2 combines the output from different SCS, and step 2.3 identifies IT integrations.

4.2.1. Step 2.1: Identification of IT systems to be replaced

In step 2.1 the IT systems or tools which will be replaced by the SCS are identified. The purpose of replacing IT systems is to standardize IT application in the specification processes [11].

4.2.2. Step 2.2: Combining output from different configuration processes

The outputs from different configuration processes in the company are combined to reduce redundancies. Communication between departments is streamlined by combining outputs and using the configuration system as a platform for data [11].

4.2.3. Step 2.3: Identification of IT integrations

Integration should be established to exchange information between the configuration system and other IT systems, such as ERP, CAD, and WMS. These IT systems can be both internal and external to the company [11].

4.3. Step 3: Establishing an overview of SCS applications

In step 3 an overview of SCS applications is created to make an initial prioritization of SCS projects. The entire specification process is mapped based on the analysis in step 1 and 2 [11]. The overview is evaluated and the SCS applications can be prioritized based on the analysis.

5. Case study: Application of the framework in a logistics company

The framework presented in the previous sections was applied to a world-leading logistics company. The

company operates 3PL warehouses for a wide range of clients and wishes to investigate how configuration systems could support the specification processes for new contracts. The services offered by the company include inbound, storage, and outbound logistics along with VAS. The warehouse flow is customized to each client and the client can request virtually any VAS. The number of offered services is almost infinite due to the broad client base and high level of customization. The company has not worked with modularization or configuration systems for warehousing services. The analysis is based on one Danish warehouse, but the results are expected to be similar in other Danish warehouses and warehouses in other European countries. After the investigation of the potential the company has started developing a prototype to test the potential of using a configurator in the sales process.

5.1. Step 1: Identifying potential SCS

5.1.1. Step 1.1: Definition of main objectives for SCS

This step provides an understanding of the objectives for implementing SCS. The goal of implementing SCS is to improve the integration of new clients. The following objectives are formulated based on interviews with stakeholders involved in the sales and integration process of new clients and the warehouse operations after implementation. The main objectives for implementing SCS in the company are as follows:

- Provide a clear split between standard and non-standard clients early in the sales process.
- Enhance understanding of the company's capabilities across departments, e.g. the type of services the company can offer.
- Improve documentation in the sales process to reduce errors and misunderstandings between departments and between the company and the client.
- Reduce lead time to generate proposals and time to integrate new clients.
- Provide a clear pricing strategy.

5.1.2. Step 1.2: Analysis of market segments

This step identifies the different market segments served by the company. Generally, the services acquired by all the clients consist of three main processes: Inbound, storage, and outbound logistics. Each process consists of several subprocesses, which are adjusted to the individual client. The client can also acquire VAS, which are services customized specifically to the client. The order of the main processes is fixed, but VAS can be added at any point in the flow. The solution space is undefined as the combination of services and number of VAS is close to infinite. Clients with very specific needs are challenging to define and require highly customized services.

Figure 2 shows the market segments, range of services, and the potential application of SCS. The case company has defined four types of clients in Denmark

based on industries. The customer segments have different demands to the main processes based on the type of goods handled. The level of customization depends on the deviation from the main processes. Clients in the highly customized segment often have very specific or demanding needs and are of strategic importance to the company. Highly standardized clients have a simple warehouse flow with few specific needs. This differentiation is defined by sales and project managers in the company. The SCS is expected to be applicable to most customer segments except for clients requiring highly customized services, e.g. significant modification to the three main processes and/or very singular VAS.

The four client types are confidential as it is not in the company's interest to disclose their clients.

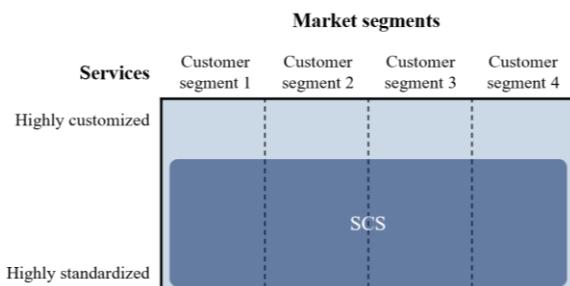


Figure 2: Potential application of SCS within the market segments.

5.1.3. Step 1.3: Identification of application areas

This step identifies application areas of SCS based on the objectives in step 1.1. The case company does not use any configurators. The analysis showed that a configurator could support the sales process and the following integration processes. Figure 3 shows the configuration processes that could be implemented and the generated specifications. The commercial configurator supports the sales department and generates lists of services with cost estimates, offers with price estimates, and drafts of contracts and standard operating procedures (SOP). When the offer is accepted by the client, information from the commercial configurator is input to the technical configurator which supports the integration processes of new clients. The technical configurator generates flow diagrams and BOM with input from project managers. The configuration system shown in Figure 1 is moderately automated as the human operators are not entirely replaced [23]. The decision to have two successive configuration stages is based on the complexity of warehousing services and the degree of service knowledge of the clients.

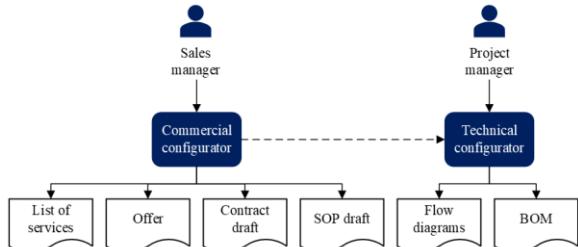


Figure 3: SCS consisting of a commercial and technical configuration process with input from two departments.

5.2. Step 2: Aligning IT development

5.2.1. Step 2.1: Identification of IT systems to be replaced

This step identifies the IT systems to be replaced with the implementation of SCS. The case company is currently using two Excel models to calculate costs and prices of services and two Word templates to generate contracts and SOPs. Based on the analysis in step 1.3, the commercial configurator can replace the Excel models and the Word templates. The remaining documents showed in Figure 1 are not supported by any specific IT systems.

As previously stated, this case study is based on the analysis of one warehouse. It was established in interviews that other warehouses are using different models and templates which are not identified in this study.

5.2.2. Step 2.2: Combining output from different configuration processes

This step combines the outputs from different configuration processes to show the information flow and dependencies across departments. Two configuration processes were identified in step 1.3. Thus, only one dependency exists between the sales department and the project managers.

5.2.3. Step 2.3: Identification of IT integrations

This step identifies integrations between the SCS and other IT systems in the company. Integration with other IT systems will be a long-term process as the company is currently not using any SCS. The configuration system should be able to retrieve data from the company's warehouse management system (WMS). This data includes key performance indicators from different warehouses and services, e.g. capacities and efficiencies. Furthermore, it would be beneficial to connect the SCS with the company's communication platform for clients, the customer relationship management (CRM) system. The communication platform is used to message the client during the sales process and exchange documents.

5.3. Step 3: Overview of SCS

In the final step an overview of the SCS is created based on step 1 and 2. The overview in Figure 4 shows the identified SCS application areas, generated output, and integration with other IT systems in the case company.

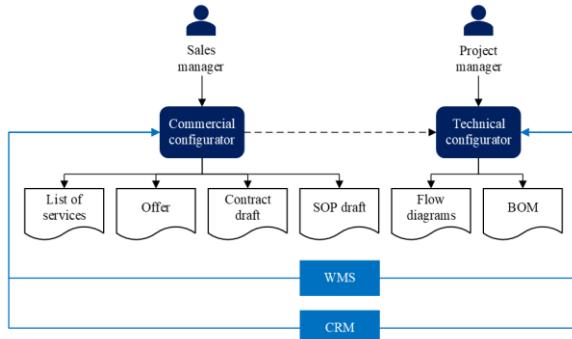


Figure 4: Overview of the potential applications of a configuration system in the case company.

The overview was used to communicate potential SCS applications in the case company. Stakeholders had a greater understanding of configuration systems and the applicability to the company's specification processes. The overview was used to make an initial prioritization of SCS projects which resulted in prioritizing the commercial configurator and generating lists of services, excluding cost estimates, for all customer segments.

The biggest challenge in the logistics service company is that each warehouse operates as its own company that develops a relationship with its clients, and its primary goal is to fulfill its client's needs. The level of customization allowed will be a challenge, as the goal of the configuration system is to standardize the different services offered in the warehouses. However, solving this by allowing customization in the commercial and technical configurator allows the warehouses to customize unique services when specific clients require them. The next step is to map the service platform of the logistics service company and investigate the level of service customization that should be allowed in the different warehouses with different customers both in type of products and size.

6. Discussion & conclusion

Logistics companies are challenged by the demand for highly customized services in 3PL warehouses. These companies must continue to meet the clients' needs to stay competitive. However, the high level of customization leads to several challenges when new clients are integrated in the warehouses.

The literature describes numerous mass customization strategies to manage similar challenges associated with products. PCS have yielded several benefits in the specification processes of customized products. However, literature on the application of configuration systems to services is limited as well as literature on modularization of services. Parallels can be drawn between logistics companies and ETO companies, which have been subject to numerous studies on implementation of PCS [7, 8, 11]. Both have an undefined

solution space and complex specification processes, which have shown to require several configurators or PCS to support the specification processes in ETO companies. Hence, it is necessary to identify and prioritize potential applications of configuration systems prior to development.

This paper contributes to literature on configuration systems for services and suggests a framework to identify potential applications of configuration systems for services. The framework provides a structured approach to analyze and scope potential projects and communicate SCS applications in organizations with no previous experience with configuration systems.

The proposed framework was based on an existing framework for ETO companies and literature within modularization of services. The framework consists of three steps: (1) Identifying potential SCS, (2) Aligning IT development, and (3) Establishing an overview of SCS applications. The main difference between the framework for products and services is the addition of a third sub-step to step 1. This step analyzes market segments to determine the applicability of configuration systems.

The framework was validated through a case study in a case company. The case company is a logistics company providing warehousing services to a wide range of clients. The company has not worked with modularization or configuration of warehousing services previously. The framework facilitated an understanding of how SCS could be applied to the case company. This was especially advantageous in identifying a need for SCS and communicating potential SCS projects to stakeholders in the organization. The case study indicated a need for analyzing market segments when identifying potential SCS due to the diverse client base. The company can use the analysis to determine future SCS projects.

The framework does not assess modularity of services in the company, which could be interesting in future studies. Investigations of modularity in services is relevant to standardize service offerings, define modules [16], and implement configuration systems [6]. Further analysis could also extend the framework and include a data-driven market segmentation with performance levels [13, 16] within the customer segments. This market segmentation could determine the exact segments and clients to include in the configuration system.

The differences between a product configuration system and a logistics service configuration system were minimal when investigating the potential application of a logistics service configuration system. However, mapping and developing the logistics service modularization is expected to be different as mapping processes that can quickly be changed and adopted differs from mapping parts of a product that will be handed over and cannot change over time.

The framework has only been applied to one case company providing logistics services. The framework should be applied to more companies to validate the approach with other types of services and different levels of customization. Likewise, the case company has yet to develop and implement the suggested SCS. The validity of the framework and the suggested application areas can be assessed once the company has developed and implemented SCS. Future research into modularization of

services and service platforms can contribute to identifying potential application areas of SCS.

Acknowledgements

The authors would like to acknowledge the case company and the stakeholders that contributed to the research.

References

- [1] G. Baruffaldi, R. Accorsi, R. Manzini, and E. Ferrari, "Warehousing process performance improvement: a tailored framework for 3PL," *Business Process Management Journal*, vol. 26, no. 6, pp. 1619–1641, Nov. 2020, doi: 10.1108/BPMJ-03-2019-0120.
- [2] Y. Lin and S. Pekkarinen, "QFD-based modular logistics service design," *Journal of Business and Industrial Marketing*, vol. 26, no. 5, pp. 344–356, Jun. 2011, doi: 10.1108/08858621111144406.
- [3] L. Jum'a and M. E. Basheer, "Analysis of Warehouse Value-Added Services Using Pareto as a Quality Tool: A Case Study of Third-Party Logistics Service Provider," *Adm Sci*, vol. 13, no. 2, Feb. 2023, doi: 10.3390/admsci13020051.
- [4] N. S. F. A. Rahman, N. H. Karim, R. M. Hanafiah, S. A. Hamid, and A. Mohammed, "Decision analysis of warehouse productivity performance indicators to enhance logistics operational efficiency," *International Journal of Productivity and Performance Management*, 2021, doi: 10.1108/IJPPM-06-2021-0373.
- [5] M. Khakdaman, J. Rezaei, and L. Tavasszy, "On the drivers of demand for innovative freight transportation services," *IEEE Engineering Management Review*, pp. 1–27, 2022, doi: 10.1109/EMR.2022.3223313.
- [6] L. Hvam, N. H. Mortensen, and J. Riis, *Product customization*. Springer, 2008. doi: 10.1007/978-3-540-71449-1.
- [7] A. Haug, L. Hvam, and N. H. Mortensen, "Definition and evaluation of product configurator development strategies," *Comput Ind*, vol. 63, no. 5, pp. 471–481, Jun. 2012, doi: 10.1016/j.compind.2012.02.001.
- [8] A. Haug, S. Shafiee, and L. Hvam, "The costs and benefits of product configuration projects in engineer-to-order companies," *Comput Ind*, vol. 105, pp. 133–142, Feb. 2019, doi: 10.1016/j.compind.2018.11.005.
- [9] G. O. Mueller, N. H. Mortensen, L. Hvam, A. Haug, and J. Johansen, "An approach for the development and implementation of commissioning service configurators in engineer-to-order companies," *Comput Ind*, vol. 142, pp. 1–16, Nov. 2022, doi: 10.1016/j.compind.2022.103717.
- [10] S. Shafiee, L. Hvam, and M. Bonev, "Scoping a product configuration project for engineer-to-order companies," *International Journal of Industrial Engineering and Management*, vol. 5, no. 4, pp. 207–220, 2014, [Online]. Available: www.ftn.uns.ac.rs/ijiem
- [11] K. Kristjansdottir, S. Shafiee, and L. Hvam, "How to identify possible applications of product configuration systems in engineer-to-order companies," *International Journal of Industrial Engineering and Management*, vol. 8, no. 3, pp. 157–165, 2017, [Online]. Available: <http://www.xx>
- [12] M. Hellström, R. Wikström, M. Gustafsson, and H. Luotola, "The value of project execution services: a problem and uncertainty perspective," *Construction Management and Economics*, vol. 34, no. 4–5, pp. 272–285, May 2016, doi: 10.1080/01446193.2016.1151062.
- [13] M. H. Meyer and A. P. Lehnard, *The power of product platforms: building value and cost leadership*. The Free Press, 1997.
- [14] S. Pekkarinen and P. Ulkuniemi, "Modularity in developing business services by platform approach," *The International Journal of Logistics Management*, vol. 19, no. 1, pp. 84–103, May 2008, doi: 10.1108/09574090810872613.
- [15] S. A. Brax, A. Bask, J. Hsuan, and C. Voss, "Service modularity and architecture – an overview and research agenda," *International Journal of Operations and Production Management*, vol. 37, no. 6, pp. 686–702, 2017, doi: 10.1108/IJOPM-03-2017-0191.
- [16] M. Løkkegaard, N. H. Mortensen, and T. C. McAloone, "Towards a framework for modular service design synthesis," *Res Eng Des*, vol. 27, no. 3, pp. 237–249, Jul. 2016, doi: 10.1007/s00163-016-0215-6.
- [17] N. Iman, "Modularity matters: a critical review and synthesis of service modularity," *International Journal of Quality and Service Sciences*, vol. 8, no. 1, pp. 38–52, Mar. 2016, doi: 10.1108/IJQSS-05-2015-0046.
- [18] T. T. Christensen, "Konfigureringsystemer for kundetilpassede serviceydelser med fokus på industrielle servicekontrakter (SLAs)," 2009.
- [19] O. Chervonnaya, "Customer role and skill trajectories in services," *International Journal of Service Industry Management*, vol. 14, no. 3, pp. 347–363, 2003, doi: 10.1108/09564230310478864.
- [20] F. Ponsignon, P. Davies, A. Smart, and R. Maull, "An in-depth case study of a modular service delivery system in a logistics context," *International Journal of Logistics Management*, vol. 32, no. 3, pp. 872–897, 2021, doi: 10.1108/IJLM-07-2020-0295.
- [21] T. Tuunanen, A. Bask, and H. Merisalo-Rantanen, "Typology for modular service design: review of literature," *International Journal of Service Science, Management, Engineering, and Technology*, vol. 3, no. 3, pp. 99–112, 2012.
- [22] C. Voss, N. Tsikriktsis, and M. Frohlich, "Case research in operations management," *International Journal of Operations and Production Management*, vol. 22, no. 2, pp. 195–219, 2002, doi: 10.1108/01443570210414329.
- [23] C. Forza and F. Salvador, *Product information management for mass customization: connecting customer, front-office and back-office for fast and efficient customization*. Palgrave Macmillan London, 2006. doi: 10.1057/9780230800922.

Multi-level configuration in smart governance systems

Salvador Muñoz-Hermoso^{1,3,*}, David Benavides^{2,3} and Francisco J. Domínguez-Mayo^{2,3}

¹Provincial Informatics Company (INPRO), Seville Provincial Council, 32 Menéndez y Pelayo St, Seville, 41071, Spain

²Dept. of Computer Languages and Systems, University of Seville, Reina Mercedes St, 41012, Seville, Spain

³DiversoLab Computer Science Laboratory, University of Seville, Reina Mercedes St, 41012, Seville, Spain

Abstract

Smart governance systems have different needs depending on the type of organization, which, together with their inherent complexity, make them difficult to configure. However, we have not found solutions that facilitate the configuration of these systems of great interest in the public sector. We propose a configurability solution compounds of a software framework-based multi-level configuration architecture, and a feature model (FM) to represent the variability in a compact way through the configuration of the different levels of the same software. Thus, the FM we present allows for obtaining a line of customized services for different organizations. On a first level, the variability of the typical collaboration processes is managed, on a second level the different collaboration models are handled, and on a third and fourth level the general smart governance system is configured, and the one adapted to the specific needs of the organization. In this way, while facilitating configuration, a high degree of accuracy is achieved regarding the specific and varying needs of the different stakeholders.

Keywords

Software Configurability, Feature Models, Software Reutilization, Smart Governance, E-Collaboration

1. Introduction

The configurability and variability management of information systems is important, in that it enables software products and services to be adapted to the needs of the organization and its stakeholders [1].

In relation to e-collaboration technologies and systems, they have an inherent complexity that is increased by the fact that their requirements and needs vary according to the application domain, and the type of organization using them [2].

In the smart governance domain (related to the broader field of e-government), the aim is to achieve quality public services and smart management (of territories and societies) through a collaborative government open to citizen, business and professional collectives, maximizing positive results with intensive use of Information Communication and Technology (ICT) [3]. Collaboration is therefore essential and particularly complex due to the different (sometimes conflicting) interests of the actors involved [4].

In the e-government exists a great variety of needs, given that governance processes and public services in a

small municipality are not the same as those in a large city, or governance in a regional or state administration [5, 6]; thus it is essential to address the different needs of citizens and governments, in terms of participation and collaboration in public policies and services. Furthermore, smart governance involves multiple stakeholders that enhances this complexity and the variety of unexpected and changing requirements, as this is still a recent field with respect to its implementation. Hence, the development of different tailor-made systems substantially increases development and configuration costs.

It is therefore desirable, to improve the reusability and address this great variety by managing the variability (which can be changeable) of a unique software or, at least, by reducing its variants and modifications. In such a complex environment, so is its configuration, thus it is also convenient to enhance the configurability of these systems. Nevertheless, we have not found any solutions in the literature review that addresses this variability facilitating moreover its configurability.

Software Product Line Engineering (SPLE) and frameworks software favor variability and reusability [6, 7, 8], and consequently, the adaptation of the software developed to the specific needs of the organization.

In this context, we propose a service-oriented configurability approach based on a multi-level software framework-based configuration architecture to provide a software product line (SPL); and a feature model (FM) in the domain of smart governance. The FM allows representing all possible configurations in a compact way [8, 9], through the configuration of the different levels of the software. So, the configuration architecture and the FM model complement each other, to facilitate the implementation of the variability of the services provided

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

✉ samu31@dipusevilla.es (S. Muñoz-Hermoso); benavides@us.es (D. Benavides); fjdominguez@us.es (F.J. Domínguez-Mayo)
🌐 http://www.lsi.us.es/~dbc/ (D. Benavides); https://www.us.es/trabaja-en-la-us/directorio/francisco-jose-dominguez-mayo (F.J. Domínguez-Mayo)

📠 0000-0002-8130-7869 (S. Muñoz-Hermoso); 0000-0002-8449-3273 (D. Benavides); 0000-0003-3502-8858 (F.J. Domínguez-Mayo)

(CC BY) © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

by these systems.

This paper is structured as follows: Section 2 explores the multi-level configuration architecture as part of the solution. In Section 3, we introduce the proposed FM to describe the configuration aspects of the smart governance system related to the presented architecture. In Section 4, a review of related works is provided to highlight the contribution of the proposed solution to previous research on SPL, e-government and e-collaboration topics. Finally, Section 5 concludes the paper, summarizing our main findings and observations, and proposing some future research works.

2. Multi-level configuration architecture

In the flat configurability approach, if a large variability of complex systems needs to be covered, multiple features and parameters must be considered, and it is difficult to manage a software product line and product customization [10].

In this section, we show a multi-level architecture proposal to support a multi-level services configuration. This approach favors configurability and reduces the complexity of managing the associated variability [10, 11, 12]. Furthermore, it is service-oriented to facilitate the development of Software-as-a-Service (SaaS) e-governance systems.

Thus, the multi-level perspective enables division and hierarchizes the configuration into different levels related to modules or system parts, requirements groups, or service models, facilitating the configuration, customization, and reuse of the services.

2.1. Framework-based architecture

In order to achieve domain adaptability, we opted for an architecture based on domain software frameworks; they favor the reuse and implementation of solutions, in particular in the public sector [7, 6, 5]. Thus, this architecture also facilitates the implementation of the configurability management of the proposed system.

To this end, we consider a first framework (*E-Collaboration Framework*) with common characteristics and functionalities for the collaboration of a group of stakeholders on certain organizational assets (documents, projects, policies, etc.). And a second framework, specialized from the first one, focused on the particular needs and services of the smart governance domain (*Smart Governance Framework*), such as citizen consultations, drafting of regulations and policies, or participation in smart city projects.

In this way, the configurability of services is addressed through the various levels of the software frameworks

and the smart governance framework-based system; in order to meet the specific needs of each organization.

2.2. Description

In Figure 1, in TOGAF-ArchiMate notation¹, we show a layered view with the most relevant artifacts of the architecture involved in the configuration, in which the different levels (gray) containing the different business objects that store the configuration are observed. This high-level modeling is suitable for representing in a visual and clear way both behavioral and static storage artifacts. The shown architecture supports the proposed FM described in the next section.

The business layer (yellow) is service-oriented and is structured in two blocks; the one on the left supports the configuration of the general e-collaboration software framework, and the block on the right offers services to manage the configuration of the adapted smart governance framework, as well as the system that is implemented around it.

2.2.1. Purpose of the levels

Since the smart governance solution is based on general domain software frameworks, it is necessary to consider a first level (0) to establish the general configuration of this reusable software in order to adapt it to the needs of the specific smart governance solution to be developed.

On the other hand, smart governance is articulated through participatory processes that are usually typified, in some cases, on the basis of citizen participation regulations. Therefore, it is appropriate to be able to define and characterize these process models through a next level (1) of configuration that will allow their adaptation (both specific and general aspects) to the needs of the organization. E.g. a certain process can be modeled for the collaborative drafting of regulations, or another for participatory budgeting. Given that there are different types of public institutions and different smart governance policies and citizen participation regulations; it is therefore desirable, to be able to model and configure them at a new configuration level (2), so that they can be reused and adapted for each institution. In addition, these models usually involve certain types of collaborative processes, hence their relationship with level 1. E.g. a model could be established for small municipalities, or for smart cities that have collaborative needs in urban projects.

¹TOGAF is an OpenGroup IEEE standard framework for developing enterprise architectures (<https://www.opengroup.org/togaf>). ArchiMate is a (graphical and semantic) modeling language for OpenGroup's high-mid level enterprise architecture under the TOGAF standard (<https://www.opengroup.org/archimate-forum/archimate-overview>).

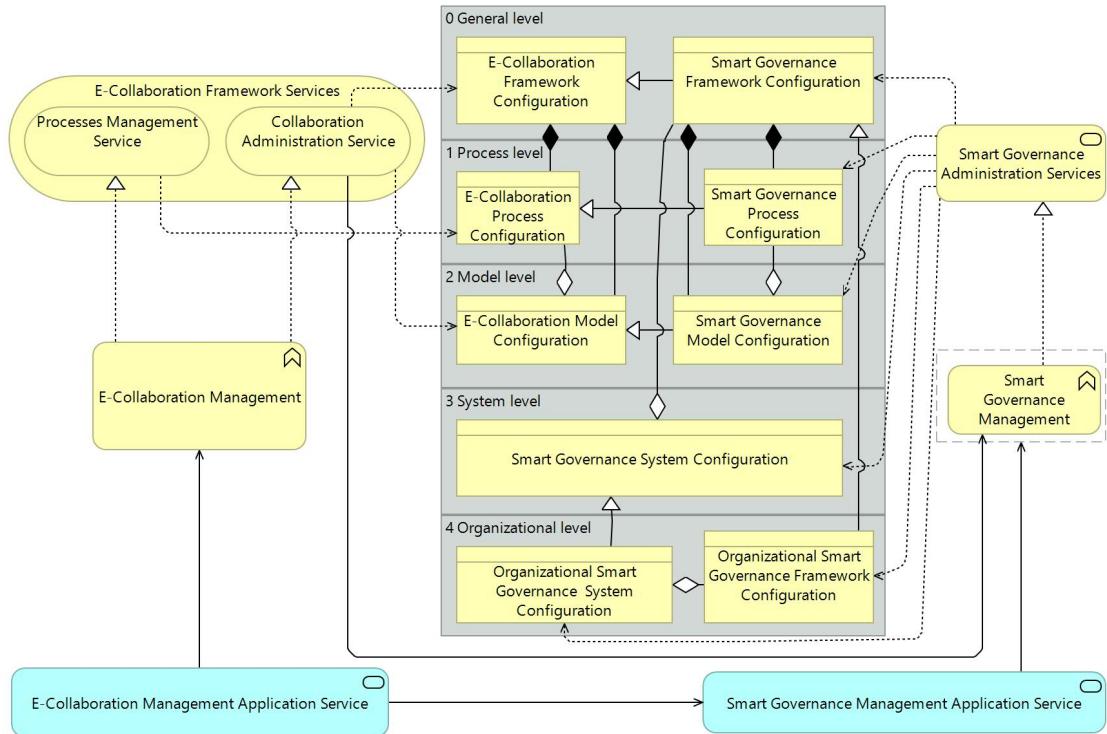


Figure 1: Architecture of the multi-level configuration of the frameworks and the system

These two levels favor dynamic variability, since enable a system in production that requires new needs, to create new types of collaborative processes or new governance models, that can be easily incorporated into the smart governance services of a specific organization, or a set of organizations using the same governance model.

While the above levels characterize the smart governance models with their associated processes in software frameworks, we need a level (3) of configuration relative to the specific system to be developed with the necessary particularities for use by specific organizations that adhere to the unique reusable system. E.g. a local administration could develop a system for use by various city councils. Furthermore, these levels make configurability more efficient, since several organizations can be customized at once, acting on levels 1 and 2 of the configuration, or on level 3 for features that globally affect the behavior of the entire system.

Based on the above, it follows that an additional level of configuration (4) is desirable, so that each organization can tailor the system to its requirements, both in terms of general and specific characteristics of the governance model and its processes. E.g. one municipality may want its participatory processes to be binding, while another may only want them to be consultative. Or regarding

specific characteristics such as the duration of certain participatory processes, participation requirements and restrictions, etc. In this way, it contributes to the precision and facility of the customization, since the configurations are inherited from top to bottom (through the levels), it is possible to customize a particular entity, only by modifying the differential features at level 4.

2.2.2. Level 0

Starting at configuration level 0 (general) and the block on the left, the *Collaboration Administration Service* facilitates the adaptation of the e-collaboration software framework through the setting of the general configuration data (*E-Collaboration Framework Configuration*), which helps to customize it in the application area. In addition, in the right-hand block, customization is enhanced by means of the *Smart Governance Administration Services*, for configuring specific features of the particular software framework in the smart governance domain (object *Smart Governance Framework Configuration*).

2.2.3. Level 1

At level 1 (processes), the *Processes Management Service* in the left-hand block allows the different types of collabora-

tive processes to be configured, storing the corresponding information in the object *E-Collaboration Process Configuration*, which is part of the framework configuration. In addition, in the right block, the *Smart Governance Administration Services* provides a finer adjustment of the processes, specifying a specific typology in the smart governance field, the configuration of which is stored in the object *Smart Governance Process Configuration*. In short, this first level manages the variability of collaborative processes, both in general and domain-specific.

2.2.4. Level 2

In order to achieve greater variability, similarly at level 2 (models), the same services of the respective frameworks facilitate the creation of a product line related to the e-collaboration and smart governance models (or organization types) that are established. The corresponding configurations are stored respectively in the objects *E-Collaboration Model Configuration* and *Smart Governance Model Configuration*.

2.2.5. Level 3

Level 3 (system), allows obtaining a specific smart governance type system, based on the framework with the desired features, in relation to the collaboration models to be supported, the types of processes, and other characteristics that were already specified in the previous configuration levels. The configuration of these features is transferred to the system and other domain-specific features are added and stored in the *Smart Governance System Configuration* object. Thus, at this level, we select a product type and a customized SPL product that provide a set of personalized smart governance services.

2.2.6. Level 4

The last level 4 (organizational), enables a configuration adapted to the specific needs of each organization adhering to the system. To this end, through the *Smart Governance Administration Services*, the system and framework configurations are inherited to be customized respectively in the objects *Organizational Smart Governance System Configuration* and *Organizational Smart Governance Framework Configuration*, which will make up the complete and specific configuration of the organization's system.

2.3. Services realization

If we go down the business layer, in the left block, we can see how the *E-Collaboration Management* functionality (functional part of the e-collaboration framework) is the one that performs the aforementioned services. Similarly,

on the right-hand side, the *Smart Governance Management* is the functionality that realizes the administration and configuration services of the system for smart governance. In terms of reusability, it relies on the general administration service of the e-collaboration framework.

Finally, in the information systems layer (light blue), the application services that support the previous business functionalities are included; *E-Collaboration Management Application Service* for the general configuration of e-collaboration, and *Smart Governance Management Application Service* for the specific configuration of smart governance services, which also relies on the previous one to enhance reuse.

3. Feature model

In this section, to complete our configurability solution, we propose a FM (Figure 2) to represent the variability of smart governance solutions with respect to the defined configuration levels, which can be implemented based on the configuration architecture of the previous section.

The FM considers the five configuration levels to obtain a family of customized products down to the specific system level of the organization.

First, we outline the key features common to the entire line of smart governance systems and, second, those that may vary for each specific system.

3.1. Common features

The system offers, on the one hand, common services for e-collaboration in any type of organization and application domain (*E-Collaboration Common Services*), and on the other hand, in the context of the study, domain-specific services (*Smart Governance Services*). These two groups are mandatory in any configuration, because these general services are necessary to accomplish any process related to smart governance. Nevertheless, these must be customized to adapt to the organization's needs, through their features and subgroups, some of them are optional.

It is also mandatory to establish the e-collaboration model to be used with its features, as well as the specific characteristics of typical collaborative processes to be used in the model. The variable cardinality of *E-Collaboration Model* and *E-Collaboration Process*, increases configurability, by making it possible to define a set of models, and for each of them, different types of processes, covering levels 1 and 2 of configuration that we discussed in the previous section; that is, different services depending on the models and types of collaborative processes that they implement. This multi-level approach to FM, in which these feature trees can be considered as

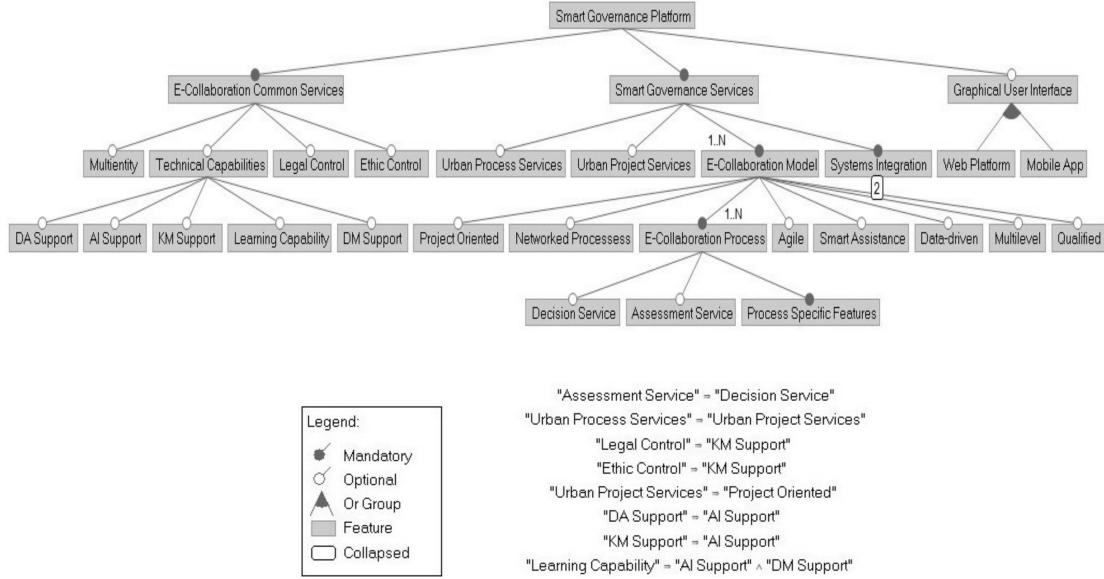


Figure 2: Feature model for a smart governance system

separate but linked FMs, allows the high complexity of these highly variable systems to be better managed [12].

In the particular application area, because of the need for information from the environment, the system cannot operate in isolation (being part of a software ecosystem), and therefore the group *System Integration* is mandatory. Within this group (in Figure 2 is collapsed), the feature *Organizational Systems & Services* is mandatory, as the system must interoperate with other existing information systems in the organization (e.g. basic citizenship data and identification services). However, we do not consider a requirement the integration with other external services (*External Global Services* feature) such as social networks or messaging services, although it would provide more information to the system.

3.2. Variable features

In relation to e-collaboration common services, a multi-entity system (*Multentity*) can be chosen, to be used independently in different entities or organizations and customized in each one of them. The features *Legal Control* and *Ethic Control*, will activate respectively the control mechanisms to favor the regulatory compliance of the domain and the organization, and its ethical values. To this end, the system must be able to model and store them like rules in a knowledge database. So these characteristics require, as we can see in Figure 2, the activation of some technical features (*Technical Capabilities*), specifically those that offer support to knowledge management (*KM Support*), to be able to handle the rules and their inference.

Those that offer support to knowledge management (*KM Support*), to be able to handle the rules and their inference.

In the field of smart governance, apart from the specific services offered, governance can be extended to urban processes and projects by activating the *Urban Process Services* and *Urban Project Services*, as the latter is required for collaboration in the former².

The group *Graphical User Interface* (GUI) is not mandatory, as an existing external interface layer can be chosen. If the system GUI is selected, a choice can be made between a web interface, a mobile interface, or both; for increased interoperability and accessibility from any device.

These features act at levels 0, 3, and 4 for entity-level customization of both the smart governance system and the software frameworks on which it is based.

In relation to level 1, we highlight within the group *E-Collaboration Process* two features that extend e-collaboration, thus the *Decision Service* will enable decision-making (DM) by the collaborators, which will optionally enable an evaluation of the same with the activation of the feature *Assessment Service*. For effective individual and collective decision-making, it is also necessary to activate technical capabilities such as the *DM Support* feature, whose group *Technical Capabilities*

²An urban process usually develops projects in the implementation phase from ideas to proposed solutions [4].

we will see at the end of the section. In smart governance, these services will enable public policy evaluation citizenship processes.

As for the optional features of the collaboration models (*E-Collaboration Model*), there are some that, with their activation, enhance the capabilities (and also the complexity) of the model: *Project-oriented* enable e-collaboration at project and project phase level (required if *Urban Project Services* are activated). *Networked-Processes* allows the creation of more complex collaborative processes based on simpler ones, or to relate processes to each other, forming a network to interoperate between them.

The *Multilevel* feature extends collaboration to the different decision-making levels of the organization (strategic, tactical, and operational), which combined with the previous feature and the *Multientity* feature, supports more complex and transversal processes in different organizations to solve common problems.

The *Agile* feature introduces an agile approach to collaboration through feedback between the different phases of e-collaboration, and even with other networked collaborative processes. This feature, together with the previous one (*Multilevel*), in the field of smart governance, facilitates dynamic collaboration throughout the public policy cycle.

The *Data-driven* feature favors collaboration and decision-making based on evidence or objective data, requiring the activation of the technical capacity *DA Support* that we will see below. Enhanced by the influence of the expertise and qualification of the collaborators (*Qualified* feature), in the decision-making process, it allows knowledge to be promoted in the final results of the decisions. Both features contribute, in smart governance, to a citizen-centric government.

As a final representative feature of the model, smart assistance (*Smart Assistance*) is envisaged, in order to support informed and effective decision-making. This requires the activation of applied AI techniques (*AI Support*) or decision-making techniques (*DM Support*).

Finally, the group of optional features *Technical Capabilities* (some of which are required by other models and processes), enhances addressing complex collaboration and decision-making problems by including technical capabilities [4] for data analysis (*DA Support*), knowledge management from relevant data (*KM Support*), individual and group knowledge-based decision-making (*DM Support*), and autonomous learning (enabled by *Learning Capability* feature), which would enable the system to autonomously configure itself to improve outcomes. It can be seen that AI techniques (*AI Support*) are required to support the above capabilities. The configuration of these features applies to levels 0, 3, and 4, enabling customization of technical capabilities at the system and organization level.

The use of variable features and the multi-level approach will not only facilitate the configuration and the customization, but also the standardization of governance processes. For example, in a context where several city councils need governance based on participatory consultations and surveys, a *E-Collaboration Model* could be defined with the desired features and with both types of processes (*E-Collaboration Process*) with their specific characteristics (*Process Specific Features*) already configured.

In this way, this model would simply have to be activated in each organization; it would also be possible to customize some of the features of the process or model in a specific entity. In addition, new process-specific features can be added to address dynamic variability more accurately, due to stakeholders' changing needs and tasks, as these are mostly related to participatory processes.

It therefore, has clear advantages over the traditional flat approach, as it would involve repeating the same configuration work over and over again for each feature and entity, and the reuse of the configuration would be more consistent, or if some standardization of processes is desired, which is desirable in public administration.

4. Related works

In the literature review, we have not found any articles that specifically address the problem of configurability in smart governance. Nor have we found proposals for multi-level configurations in the more general domain of e-government.

However, we have identified some e-government studies that although they do not focus on improving configurability, do address how to facilitate the development and adaptation of these e-government systems to different needs through Software Product Line (SPL) and domain frameworks [13, 6, 14]. And others like in [15] propose using Feature Models (FM). This work is interesting, because of its broad vision as ours, since proposes a general model for e-governance systems; furthermore, they establish a division by front-office or back-office software, and another by applications typology: Government to Government (G2G), Government to Business (G2B), or Government to Citizen (G2C).

In [6] further distinguish products for central or local governments, which is appropriate as the latter offers public services related to city government; quite different from those offered by the state. Regarding our work, adaptation to a local, regional, or central government, could be accomplished through different models defined in level 2 (models) of configuration.

In [5] they also propose a framework approach but do not address configurability as a specific problem, but focus more on facilitating the development of electronic

public services. SPL is also applied in some particular use cases such as the one proposed in [16] for content management systems (CMS).

In short, as in [13] is mentioned, there are few studies that address variability and SPL in the e-government domain, so this is an area that needs to be explored further.

5. Conclusion and future works

The Feature-Model (FM) and the architecture that supports it, proposed in this work, facilitate multi-level service-oriented configurability (at the level of the general e-collaboration software framework and its processes, the particularized framework in the domain, and the smart governance system), product line configurability (each model can be considered a product for a particular type of smart governance or institution), and multi-entity configurability (supporting different configurations for each organization). Therefore, from a single software system, through reuse and configuration, it is possible to obtain a significant dynamic variability of services for e-collaboration and in particular for smart governance. Moreover, the *Learning Capability* and *AI Support* features will enable an autonomous configuration to evolve the system towards configurations more adapted to the organizations.

Concerning other related proposals, ours focuses on the specific problem of configurability from a general perspective by providing several complementary methods and techniques integrated into the solution: multi-level configuration architecture, domain software frameworks, SPL and FM; as well as TOGAF-Archimate as formal modeling framework.

The preliminary results show that the configurability architecture proposed in the present study contributes to the general area of e-collaboration, and in particular of smart governance, to facilitate the characterization and configuration of these systems, also favoring their reusability, and adaptability with respect to the particular and varying needs of the different stakeholders and organizations.

Since we have not carried out a systematic review of other possible configurability solutions in other areas, a follow-up to this work could be to conduct such a study to establish possible relations and synergies. Another future work could be envisaged to further specify the configuration architecture and the FM, aimed at developing software prototypes, either in general, in the specific domain of smart governance, or another application area.

The development of a prototype for a given use case (e.g. a governance system for a specific city council) would help to validate our contribution. To this end, quantitative performance metrics (e.g. related to the time spent on configuration processes, its complexity, or ac-

curacy) could also be studied to empirically assess its efficiency and effectiveness compared to other approaches and proposals to manage configurability.

Furthermore, the FM could also be specified at a higher level of detail by developing feature sets, e.g. the *Process Specific Features*, or exploring new features that may be of interest. Tools to support the proposal would also be of interest, e.g. to validate the consistency of the model in relation to the features that are selected, as well as to generate the corresponding software services from them.

Acknowledgments

This work was supported by the *VII Own Plan* research aid from the University of Seville, FEDER/Ministry of Science and Innovation/Junta de Andalucía/State Research Agency with the following grants: *Data-pl*(PID2022-138486OB-I00) , TASOVA PLUS research network (RED2022-134337-T) and *METAMORFOSIS* (FEDER_US-1381375)

References

- [1] J. Bosch, Software variability management, in: Proceedings - International Conference on Software Engineering, volume 26, Elsevier B.V, 2004, pp. 720–721. doi:10.1016/j.scico.2004.06.001.
- [2] B. E. Munkvold, I. Zigurs, Integration of e-collaboration technologies: Research opportunities and challenges, International Journal of e-Collaboration (IJeC) 1 (2005) 1–24.
- [3] C. Jiménez, Una aproximación al concepto de Gobernanza Inteligente., Perspectiva (2013) 44–48.
- [4] G. Tran Thi Hoang, L. Dupont, M. Camargo, Application of Decision-Making Methods in Smart City Projects: A Systematic Literature Review, Smart Cities 2 (2019) 433–452. URL: <https://www.mdpi.com/2624-6511/2/3/27>. doi:10.3390/smartcities2030027.
- [5] A. Ojo, T. Janowski, E. Estevez, A Composite Domain Framework for Developing Electronic Public Services. (2007). URL: <https://mural.maynoothuniversity.ie/15885/>.
- [6] A. N. Fajar, I. M. Shofi, Reduced software complexity for E-Government applications with ZEF framework, Telkomnika (Telecommunication Computing Electronics and Control) 15 (2017) 415–420. doi:10.12928/TELKOMNIKA.v15i1.3195.
- [7] I. Achour, L. Labed, R. Helali, H. B. Ghazela, A Service Oriented Product Line Architecture for E-Government, The 2011 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government, EEE 2011 (2011) 186 – 192.

- [8] K. Pohl, G. Böckle, F. Van Der Linden, Software product line engineering: Foundations, principles, and techniques, Springer Berlin / Heidelberg, Berlin, Heidelberg, 2005. doi:[10.1007/3-540-28901-1](https://doi.org/10.1007/3-540-28901-1).
- [9] D. Benavides, S. Segura, A. Ruiz-Cortés, Automated analysis of feature models 20 years later: A literature review, in: Actas de las 16th Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2011, 2011, pp. 951–952.
- [10] T. Clark, U. Frank, I. Reinhartz-Berger, A. Sturm, A multi-level approach for supporting configurations: A new perspective on software product line engineering (2017).
- [11] K. Czarnecki, S. Helsen, U. Eisenecker, Staged configuration through specialization and multilevel configuration of feature models, Software process: improvement and practice 10 (2005) 143–169.
- [12] M. O. Reiser, M. Weber, Multi-level feature trees: A pragmatic approach to managing highly complex product families, in: Requirements Engineering, volume 12, Springer Nature, NEW YORK, 2007, pp. 57–75. doi:[10.1007/s00766-007-0046-0](https://doi.org/10.1007/s00766-007-0046-0).
- [13] G. Cledou, L. S. Barbosa, Modeling families of public licensing services: A case study, in: Proceedings - 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering, FormaliSE 2017, IEEE, 2017, pp. 37–43. doi:[10.1109/FormaliSE.2017.8](https://doi.org/10.1109/FormaliSE.2017.8).
- [14] I. Achour, L. Labed, R. Helali, H. B. Ghazela, A Service Oriented Product Line Architecture for E-Government, in: The 2011 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government, EEE 2011, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Athens, 2011, pp. 186 – 192.
- [15] N. Debnath, L. Felice, G. Montejano, D. Riesco, A feature model of E-government systems integrated with formal specifications, Proceedings - International Conference on Information Technology: New Generations, ITNG 2008 (2008) 27–31. doi:[10.1109/ITNG.2008.104](https://doi.org/10.1109/ITNG.2008.104).
- [16] V. M. A. D. Lima, R. M. Marcacini, M. H. P. Lima, M. I. Cagnin, M. A. S. Turine, A generation environment for front-end layer in e-government content management systems, in: Proceedings - 9th Latin American Web Congress, LA-WEB 2014, IEEE, 2014, pp. 119–123. doi:[10.1109/LAWeb.2014.20](https://doi.org/10.1109/LAWeb.2014.20).

Dynamic Aggregates in Expressive ASP Heuristics for Configuration Problems

Richard Comploi-Taupe^{1,*}, Gerhard Friedrich^{2,*} and Tilman Niestroj^{2,*}

¹Siemens AG Österreich, Vienna, Austria

²Universität Klagenfurt, Klagenfurt, Austria

Abstract

First-order logic has been applied successfully to real-world configuration problems through Answer Set Programming (ASP). To extend the application scope of ASP, lazy grounding and domain-specific heuristics were introduced. Domain-specific heuristics support the problem solver in selecting choices aiming at minimizing the search effort. Dynamic heuristics exploit the current state of the problem-solving process and assign priorities to choices. Depending on the domain, heuristics must be formulated which reason about the properties of sets of atoms. E.g., how many components are connected to a particular type of component, or what is the current sum/maximum/minimum of a physical quantity (power, voltage, current, etc.) of a particular subconfiguration? For expressing such queries, ASP offers aggregates. The semantics of these aggregates are defined w.r.t. a complete solution. However, in dynamic heuristics, the problem solver has to reason about partial solving states. In this paper, we extend heuristics in ASP with dynamic aggregates and show their implementation as well as effectiveness.

Keywords

knowledge-based configuration, answer set programming, heuristics

1. Introduction

Answer Set Programming (ASP) [1] is a declarative knowledge representation and reasoning framework based on first-order logic that has been applied successfully to a variety of industrial problems [2] such as configuration [3]. Current ASP solvers transform first-order descriptions of problem instances into propositional logic (called grounding) and apply a propositional problem solver (e.g., backtracking search) to generate solutions. However, applications manifested two issues with the ground-and-solve approach. The first issue is the so-called *grounding bottleneck*: Large problem instances cannot be grounded by modern grounders like gringo [4] in acceptable time and space. The second issue is that, even if the problem can be grounded, computation of answer sets might take considerable time, as indicated by ASP competition reports [5].

Both issues were recently addressed. First, to overcome the grounding bottleneck, lazy grounding ASP systems interleave grounding and solving to instantiate and store only relevant parts of the ground program in memory. The second performance-related issue is tackled by modern solvers using various techniques, among which domain-specific heuristics play a central role.

The work in [6] extends existing approaches by (1) in-

troducing dynamic heuristics and (2) their exploitation in a lazy-grounding ASP system. The ASP system extended by this approach is Alpha [7], the most actively developed lazy-grounding system available. Dynamic heuristics allow reasoning about the current problem-solving state represented by a partial assignment of truth values to some (but not all) atoms of a logical specification.

This reasoning may require the application of aggregation. E.g., during the configuration process of electronic equipment, an effective heuristic for problem-solving can say: Given the current state of problem-solving, select the most power-hungry, currently unconnected electronic board, and connect this board to the rack with minimal power consumption (i.e., the rack for which the total power consumption of all boards currently connected is minimal).

However, state-of-the-art ASP aggregates are evaluated only w.r.t. a complete assignment of truth values, i.e., only if every atom (proposition) is true or false, so that their value cannot change during solving. For our rack configuration example, this semantics implies that the power consumption of a rack can only be determined if the assignment of boards to a rack is final. Consequently, ASP aggregates like sum cannot be employed in dynamic heuristics to reason about the current search state where board assignments to a rack are not completed.

In this paper, we introduce *dynamic aggregates*, which are computed w.r.t. the current state of problem-solving. Consequently, such aggregates can be exploited in dynamic heuristics to steer the reasoning process depending on the current state of problem-solving.

The paper is organized as follows. In Section 2, we give a brief introduction to ASP and lazy grounding. Section 3

ConfWS’23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Authors are listed in alphabetic order.

✉ richard.taupe@siemens.com (R. Comploi-Taupe); gerhard.friedrich@aau.at (G. Friedrich); tilmanni@edu.aau.at (T. Niestroj)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

provides a driving example and introduces dynamic aggregates informally. In Section 4, we present the syntax and semantics of dynamic aggregates. Section 5 shows the implementation and integration of dynamic aggregates using a query-driven approach. Finally, in Section 6, we present the results of our evaluation.

2. Answer Set Programming

Answer Set Programming (ASP) [1] is an approach to declarative programming. Instead of stating how to solve a problem, the programmer formulates the problem as a logic program specifying the search space and the properties of valid solutions. An ASP solver then finds models (so-called *answer sets*) for this logic program, which correspond to solutions for the original problem.

2.1. Syntax

ASP offers a rich input language, of which we introduce only the core concepts needed in this paper. For a comprehensive definition of ASP's syntax and semantics, we refer to [8].

Let $\langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ define a first-order language, where \mathcal{V} is a set of variable symbols, \mathcal{C} is a set of constant symbols, \mathcal{F} is a set of function symbols, and \mathcal{P} is a set of predicate symbols.

A classical *atom* is of the form $p(t_1, \dots, t_n)$, where $p \in \mathcal{P}$ is a predicate symbol and t_1, \dots, t_n are *terms*. Each variable $v \in \mathcal{V}$ and each constant $c \in \mathcal{C}$ is a term. Furthermore, for $f \in \mathcal{F}$, $f(t_1, \dots, t_n)$ is a function term. ASP also allows built-in atoms, such as equality or comparison predicates, which take arithmetic terms as arguments, e.g., $X**2 > 1$ where $**$ is the power operator.

An answer-set program P is a finite set of rules of the form

$$h \leftarrow b_1, \dots, b_m, \text{not } b_{m+1}, \dots, \text{not } b_n. \quad \langle 1 \rangle$$

where h and b_1, \dots, b_n are atoms and *not* is negation as failure (a.k.a. default negation), which refers to the absence of information, i.e., an atom is assumed to be false as long as it is not derived by some rule. A *literal* is either an atom a or its negation $\text{not } a$. Given a rule r of the form $\langle 1 \rangle$, $\text{head}(r) = \{h\}$ is called the *head* of r , and $\text{body}(r) = \{b_1, \dots, b_m, \text{not } b_{m+1}, \dots, \text{not } b_n\}$ is called the *body* of r . By $\text{body}^+(r) = \{b_1, \dots, b_m\}$ and $\text{body}^-(r) = \{b_{m+1}, \dots, b_n\}$ we denote the positive and negative atoms in the body of r , respectively. A rule r where $\text{head}(r) = \emptyset$, e.g., $\leftarrow b$, is called *constraint*. A rule r where $\text{body}(r) = \emptyset$, e.g., $h \leftarrow .$, is called *fact*. In facts the arrow can be omitted. A rule is ground if all its atoms are variable-free. A ground program comprises only ground rules.

2.2. Semantics

Given a program P , the *Herbrand universe* of P , denoted by U_P , consists of all integers and of all ground terms constructible from constant symbols and function symbols appearing in P . The *Herbrand base* of P , denoted by B_P , is the set of all ground classical atoms that can be built by combining predicates appearing in P with terms from U_P as arguments [8].

A substitution σ is a mapping from variables \mathcal{V} to elements of the Herbrand universe U_P of a program P . Let O be a rule, an atom, or a literal, then by $O\sigma$ we denote a rule, atom, or literal obtained by replacing each variable $v \in \text{vars}(O)$ by $\sigma(v)$. The function vars maps any rule, atom, literal, or any other object containing variables to the set of variables it contains. For instance, $\text{vars}(a(X)) = \{X\}$ and for a rule $r_1 : a(X) \leftarrow b(X, Y)$, $\text{vars}(r_1) = \{X, Y\}$.

As usual, we assume rules to be *safe*, which is the case for a rule r if $\text{vars}(r) \subseteq \bigcup_{a \in \text{body}^+(r)} \text{vars}(a)$, e.g., all variables must occur in the positive atoms of the rule, which allows the grounding process to substitute them with constants.

The (ground) instantiation of a rule r equals $r\sigma$ for some substitution σ , which maps all variables in r to ground terms. The (ground) instantiation $\text{grd}(P)$ of a program P is the set of all possible instantiations of the rules in P [8]. Function symbols may cause the Herbrand base and the full grounding of a program to be infinite. By restricted usage of function symbols, answer-set programs can be designed in a way that reasoning is decidable.

An *Herbrand interpretation* for a program P is a set of ground classical atoms $I \subseteq B_P$. A ground classical atom a is true w.r.t. an interpretation I , denoted $I \models a$, iff $a \in I$. A ground literal $\text{not } a$ is true w.r.t. an interpretation I , denoted $I \not\models a$, iff $I \not\models a$. A ground rule r is *satisfied* w.r.t. I , denoted $I \models r$, if its head atom is true w.r.t. I ($h \in \text{head}(r) : I \models h$) whenever all body literals are true w.r.t. I ($\forall b \in \text{body}(r) : I \models b$). An interpretation I is a model of P , denoted $I \models P$, if $I \models r$ for all rules $r \in \text{grd}(P)$.

Given a ground program P and an interpretation I , let P^I denote the transformed program obtained from P by deleting rules in which a body literal is false w.r.t. I : $P^I = \{r \mid r \in P, \forall b \in \text{body}(r) : I \models b\}$.

An interpretation I of a program P is an *answer set* of P if it is a subset-minimal model of $\text{grd}(P)^I$, i.e., I is a model of $\text{grd}(P)^I$ and there exists no $I' \subsetneq I$ that is a model of $\text{grd}(P)^I$.

2.3. Notation

In this section, we introduce some notation that will be used later in the article.

An *assignment* A over B_P is a set of signed literals $\mathbf{T} a$, $\mathbf{F} a$, or $\mathbf{M} a$, where $\mathbf{T} a$ and $\mathbf{F} a$ express that an atom a

is true and false, respectively, and $\mathbf{M} a$ indicates that a “must-be-true”. \mathbf{M} means that an atom must eventually become true by derivation in a correct solution extending the current partial assignment, but no derivation has yet been found that would make the atom true. E.g., given constraint $\leftarrow \text{not } b$. we know that atom b must be true and has to eventually become true by derivation. Intuitively, $\mathbf{T} b \in A$ means that b is true and justified, i.e., derived by a rule that fires under A , while $\mathbf{M} b \in A$ only indicates that b is true but potentially not derived. Let $A_s = \{a \mid s a \in A\}$ for $s \in \{\mathbf{F}, \mathbf{M}, \mathbf{T}\}$ denote the set of atoms occurring with a specific sign in assignment A . We assume assignments to be *consistent*, i.e., no negative literal may also occur positively ($A_{\mathbf{F}} \cap (A_{\mathbf{M}} \cup A_{\mathbf{T}}) = \emptyset$), and every positive literal must also occur with must-be-true ($A_{\mathbf{T}} \subseteq A_{\mathbf{M}}$). The latter condition ensures that assignments are monotonically growing (w.r.t. set inclusion) in case an atom that was must-be-true becomes justified by a rule deriving it and hence changes to true.

An assignment A is *complete* if every atom in the Herbrand base is assigned true or false ($\forall a \in B_P : a \in A_{\mathbf{F}} \cup A_{\mathbf{T}}$). An assignment that is not complete is *partial*.

Many useful language constructs have been introduced to extend the basic language of ASP defined in Sections 2.1 and 2.2. We discuss such extensions only briefly and refer to [8] and [1] for full details.

A *cardinality atom* is of the form

$$lb \{a_1 : l_{1_1}, \dots, l_{m_1}; \dots; a_n : l_{1_n}, \dots, l_{m_n}\} ub,$$

where, for $1 \leq i \leq n$, $a_i : l_{1_i}, \dots, l_{m_i}$ represents a *conditional literal* in which a_i (the head of the conditional literal) is a classical atom and all l_j are literals, and lb and ub are integer terms indicating a lower and an upper bound, respectively. If one or both of the bounds are not given, their defaults are used, i.e., 0 for lb and ∞ for ub . A cardinality atom is satisfied if $lb \leq |C| \leq ub$ holds, where C is the set of head atoms in the cardinality atom that are satisfied together with their conditions (e.g., l_{1_i}, \dots, l_{m_i} for a_i).

As an extension of cardinality atoms, ASP also supports aggregate atoms that apply aggregate functions like *count* or *sum* to sets of literals. An aggregate atom is satisfied if the value computed by the aggregate function respects the given bounds, e.g., $1 = \#\text{sum}\{1 : a; 2 : b\}$ is satisfied if a but not b is true.

2.4. Lazy Grounding

Lazy grounding is an approach that interleaves the solving and grounding phases, such that computations are guaranteed to yield all answer sets. The foundation for lazy grounding is known as the computation sequence [9]. A computation sequence $S = \langle S_0, S_1, \dots, S_n \rangle$ is a sequence of partial assignments that is monotonically growing (w.r.t. set inclusion). Every element S_i of the sequence

represents the state of the computation at step i . The first element of the sequence is empty ($S_0 = \emptyset$), and every other element S_i contains the signed literals that can be derived from the preceding partial assignment S_{i-1} in the program P .

Since each element of a computation sequence is a partial assignment containing signed literals, and the sequence is monotonically growing, each S_i contains atoms assigned \mathbf{T} that will remain true in all extensions of S_i , and atoms assigned \mathbf{F} that will definitely remain false in all extensions of S_i .

Computation sequences require a normal logic program as input (i.e., rules of the form $\langle 1 \rangle$ without cardinality atoms and aggregate atoms, cf. [9, 10, 7]). Hence lazy grounding systems usually only accept normal logic programs or, in the case of Alpha, rewrite enhanced ASP constructs like aggregates into normal rules.

A rule r is said to be *applicable* in S_i if $\{\mathbf{T} a \mid a \in \text{body}^+(r)\} \subseteq S_i$ and $\{\mathbf{M} a \mid a \in \text{body}^-(r)\} \cap S_i = \emptyset$, i.e., if the positive body is satisfied and S_i does not contradict the negative body. For every applicable rule r in S_i without a negative body, the partial assignment S_i is extended to S_{i+1} by $\mathbf{T} \text{head}(r)$.

Based on the fact that the computation sequence only needs to know those ground rules that are applicable only those rules are grounded, whose positive body holds in the current partial assignment.

Each applicable rule r in S_i with a non-empty negative body constitutes an *active choice point*. Given a set of choice points for S_i the problem solver has to decide which rule to apply. Applying an applicable rule r has the consequence that S_i is extended to S_{i+1} by adding $\mathbf{T} \text{head}(r)$ and $\mathbf{F} a$ for all $a \in \text{body}^-(r)$, i.e., all atoms of the negative body are assumed to be false.

In the following example in S_0 , Rule 1 is the only applicable rule. Consequently, $S_1 = \{\mathbf{M} x(1), \mathbf{T} x(1)\}$. In S_1 Rules 2 and 3 are applicable. If the solver decides to apply Rule 2 then $\mathbf{M} b(1)$, $\mathbf{T} b(1)$ and $\mathbf{F} c(1)$ are added to assignment S_2 and therefore Rule 3 is not applicable in S_2 .

$x(1) \leftarrow .$	%	Rule 1
$b(1) \leftarrow x(1), \text{not } c(1).$	% guessing b	Rule 2
$c(1) \leftarrow x(1), \text{not } b(1).$	% guessing c	Rule 3

Deciding which rule to apply is based on heuristics which may be general, i.e., designed for every ASP program [11], or they may be domain-specific, e.g., designed for a specific problem [6].

3. Example

As an introductory example, consider the following ASP program. The idea is that for every number $i \in \{1, \dots, n\}$ the solver can decide either to assert $b(i)$ or $c(i)$. As an

example we set $n = 400$. Let B_s and C_s be all the $b/1$ and $c/1$ atoms in an answer set of the example program. We require that any answer set must fulfill the constraint $((\sum_{b(i) \in B_s} i) - (\sum_{c(i) \in C_s} i))^2 \leq 1$, e.g., the difference between these two sums must be at most 1. We call this problem the *Balanced Sum Problem (BSP)*. The example program comprises a guessing part and a part where solutions are checked. Moreover, we may specify initial facts like $b(200)$ and $b(201)$. In the worst case, 2^{398} guesses are possible. To avoid a high number of possible guesses, we can formulate heuristics that aid the solver in performing correct guesses such that backtracking is minimized.

```

x(1..400).           % initializing values from 1 to 400.
% guessing
b(X) ← x(X), not c(X).          % guessing b
c(X) ← x(X), not b(X).          % guessing c
% initial imbalance
b(200). b(201).
% check solution
sumB(Sum) ← Sum = #sum{Y : b(Y)}.
sumC(Sum) ← Sum = #sum{Y : c(Y)}.
% constrain difference between sums
← sumB(SB), sumC(SC), (SB - SC)**2 > 1.
% heuristics
#heuristic b(X) :                % b-heuristic
  x(X), not c(X), S = #sum{Y : c(Y)},
  Weight = X, Level = S. [Weight@Level]
#heuristic c(X) :                % c-heuristic
  x(X), not b(X), S = #sum{Y : b(Y)},
  Weight = X, Level = S. [Weight@Level]

```

As an example, let us consider an instantiated version of a heuristic for the partial assignment $S_1 = \{\mathbf{M} b(200), \mathbf{T} b(200), \mathbf{M} b(201), \mathbf{T} b(201), \mathbf{M} x(1), \mathbf{T} x(1), \dots, \mathbf{M} x(400), \mathbf{T} x(400)\}$, i.e., the partial assignment comprising all initial facts which are true. For $x(400)$ an instance of the c-heuristic (including the evaluation of the aggregate) is $\#\text{heuristic } c(400) : x(400), \text{not } b(400), 401 = \#\text{sum}\{200, 201\}, [400@401]$.

Heuristic directives assign a weight and a level to a rule which derives an atom. In this instantiated heuristic directive, the weight is 400, and the level is 401. All other instantiated c-heuristics and b-heuristics have either a lower level or lower weights in case of the same level. For performing choices, guesses are preferred with higher levels, and higher weights are prioritized among guesses with the same level. Consequently, the solver will apply a rule which asserts $c(400)$.

The novel concept of this paper is that aggregates in heuristic directives like $\#\text{sum}$ are evaluated w.r.t. the current assignment. For the partial assignment S_1 , the aggregate $\#\text{sum}\{Y : b(Y)\}$ in the c-heuristic is evaluated as $\#\text{sum}\{200, 201\}$ since S_1 contains the atoms $\mathbf{T} b(200)$ and $\mathbf{T} b(201)$. Applying the aggregate function $\#\text{sum}$ derives 401. Note, in the partial assignment S_1 , the c-heuristic is

not applicable if the $\#\text{sum}$ aggregate is evaluated under the standard declarative semantics of ASP. This semantics assumes that the truth assignments for the $b/1$ atoms are fixed.

By adding the shown heuristic directives to the example program, wrong choices, which lead to backtracks, can be avoided for the depicted problem instance. The following section will present the syntax and semantics of heuristics that employ dynamic aggregates.

4. Syntax and semantics

In [6] domain-specific heuristics for answer set programming were proposed which allow to reason about the current state of the problem-solving process. This state is reflected by the latest partial assignment. Consequently, heuristic directives are evaluated w.r.t. this assignment. However, in the declarative semantics of ASP the truth value of aggregates as presented in the example (e.g., $S = \#\text{sum}\{Y : b(Y)\}$) can only be determined w.r.t. a fixed set of truth assignments for atoms. In the declarative semantics of ASP assigning a truth value to $S = \#\text{sum}\{Y : b(Y)\}$ implies that the set of $b/1$ atoms which are assigned to true is fixed, i.e., rules must not be applied which assert additional $b/1$ atoms to true.

However, in the spirit of [6] we propose to evaluate aggregates w.r.t. the latest partial assignment S_i to evaluate heuristic directives for determining the choice, i.e., which rule to apply to compute the next partial assignment.

Definition 1 (Heuristic Directive). A heuristic directive is of the form $\langle 2 \rangle$, where a_i ($0 \leq i \leq n$) are atoms and w and l are integer terms.

```
#heuristic a0 : a1, ..., ak,
                     not ak+1, ..., not an. [w@l]  ⟨2⟩
```

The heuristics' head is given by a_0 and its condition by $\{a_1, \dots, a_k, \text{not } a_{k+1}, \dots, \text{not } a_n\}$.

We call an atom in a heuristic directive a *heuristic atom*. We now describe our semantics, beginning with the condition under which a heuristic atom is satisfied.

Definition 2 (Satisfying a Heuristic Atom). Given a ground heuristic atom a and a partial assignment A , a is satisfied w.r.t. A iff $a \in A_T$, i.e., atom a is assigned to true.

The *head* of a heuristic directive d of the form $\langle 2 \rangle$ is denoted by $\text{head}(d) = a_0$, its *weight* by $\text{weight}(d) = w$ if given, else 0, and its *level* by $\text{level}(d) = l$ if given, else 0. The *(heuristic) condition* of a heuristic directive d is denoted by $\text{cond}(d) := \{a_1, \dots, a_k, \text{not } a_{k+1}, \dots, \text{not } a_n\}$, the *positive condition* is $\text{cond}^+(d) := \{a_1, \dots, a_k\}$ and the *negative condition* is $\text{cond}^-(d) := \{a_{k+1}, \dots, a_n\}$.

Whether a heuristic directive is satisfied depends on whether the atoms occurring in the directive are satisfied.

Definition 3 (Satisfying a Heuristic Directive).

Given a ground heuristic directive d and a partial assignment A , $\text{cond}(d)$ is satisfied w.r.t. A iff: every $a \in \text{cond}^+(d)$ is satisfied and no $a \in \text{cond}^-(d)$ is satisfied.

Intuitively, a heuristic condition is satisfied iff its positive part is fully satisfied and none of its default-negated literals is contradicted.

Definition 4 (Applicability of a Heuristic Directive).

A ground heuristic directive d is applicable w.r.t. a partial assignment A and a ground program P iff: $\text{cond}(d)$ is satisfied, $\exists r \in P$ s.t. $\text{head}(r) = \text{head}(d)$ and $\{\mathbf{T} a \mid a \in \text{body}^+(r)\} \subseteq A$ and $\{\mathbf{M} a \mid a \in \text{body}^-(r)\} \cap A = \emptyset$, and $\text{head}(d) \notin (A_{\mathbf{T}} \cup A_{\mathbf{F}})$.

Intuitively, a heuristic directive is applicable iff its condition is satisfied, there exists a currently applicable rule that can derive the atom in the heuristic directive's head, and the atom in its head is assigned neither \mathbf{T} nor \mathbf{F} . If the atom in the head is assigned \mathbf{M} , the heuristic directive is still applicable, because any atom with the non-final truth value \mathbf{M} must be either \mathbf{T} or \mathbf{F} in any answer set.

What remains to be defined is the semantics of weight and level. Given a set of applicable heuristic directives, one directive with the highest weight will be chosen from the highest level.

Definition 5 (Heuristics Eligible for Choice).

Given a set D of applicable ground heuristic directives, the subset eligible for immediate choice is defined as $\text{maxpriority}(D)$ in two steps:

$$\begin{aligned} \text{maxlevel}(D) &:= \{d \mid d \in D \text{ and} \\ &\quad \text{level}(d) = \max_{d \in D} \text{level}(d)\} \\ \text{maxpriority}(D) &:= \{d \mid d \in \text{maxlevel}(D) \text{ and} \\ &\quad \text{weight}(d) = \max_{d \in \text{maxlevel}(D)} \text{weight}(d)\} \end{aligned}$$

After choosing a heuristic using maxpriority , a solver makes a decision on the directive's head atom. Other solving procedures, e.g., deterministic propagation, are unaffected by processing heuristics. In case no heuristic directive is applicable or multiple directives have the same maxpriority the solver's default heuristic (e.g., VSIDS) makes a choice as usual.

Aggregate atoms may be employed in the condition of a heuristics directive. An *aggregate atom* is of the form

$$s_1 \prec_1 \alpha \{t : l_1, \dots, l_m; \dots; \\ t : l_n, \dots, l_m\} \prec_2 s_2$$

where t corresponds to a variable, an integer, or a ground atom. We call t an aggregate term. α refers to some aggregate function that is applied to the multiset of aggregate terms t that remain after evaluating the condition

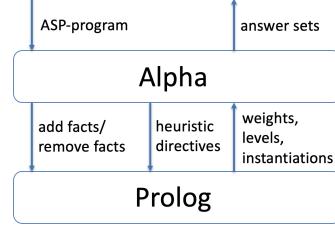


Figure 1: Integration of Alpha and Prolog

l_1, \dots, l_m . The aggregate terms are treated as members of a multiset. Duplicates are allowed.¹

The result of applying α is exploited to evaluate the comparison condition expressed by $s_1 \prec_1$ and $\prec_2 s_2$. These conditions may be omitted. s_1, s_2 are terms, e.g., numbers or variables. $s_1 \prec_1$ and $\prec_2 s_2$ are called guards. For the guard operator \prec comparison operators such as $=, \neq, \leq, \geq, <, >$ may be employed.

If in $t : l_1, \dots, l_m$ of an aggregate atom the term t is a variable, then this variable must be safe. This variable is safe if it is contained in the condition or it is a global variable. A variable in a heuristic directive d is global if it appears in a classical atom in $\text{cond}^+(d)$ or in a guard of an aggregate atom of d where \prec corresponds to $=$.

We allow aggregate functions α like $\#count$ (the number of aggregate terms) or $\#\sum$ (sum of aggregate terms).

An aggregate atom is satisfied if the value computed by the aggregate function respects the given bounds, e.g., $1 = \#\sum\{1 : a; 1 : b\}$ is satisfied if either a or b is true. Let us assume that the facts $a(1), a(2), a(5)$ are given. Evaluating the aggregate atom $X = \#\sum\{Y : a(Y); Y : b(Y)\}$ will bind 8 to variable X .

5. Integration into a lazy-grounding ASP solver

In search for answer sets, Alpha applies heuristics to select an active choice point. In contrast to [6], the heuristic directives are transformed to Prolog queries and evaluated by a Prolog interpreter. We have chosen this approach to implement the efficient evaluation of dynamic aggregates in heuristic directives.

Figure 1 shows the integration of Alpha with Prolog.

Query-driven heuristics are employed by Alpha if the $-uqh$ flag is set. The heuristic directives are removed from the input program and translated into internal data structures. These data structures comprise all the necessary

¹Note that this semantics differs from the ASP semantics of aggregates employed in rules. First, for our prototypical system t is a single term and not a tuple of terms for simplicity reasons. Second, we allow a multiset of aggregate terms instead of a set, i.e., we do not remove duplicates. Sets and multisets can be easily implemented. However, the removal of duplicates may introduce additional computational costs.

information for evaluating the heuristic directives, such as their head atom, variables, atoms occurring inside the heuristic, and crucially, their respective Prolog query. Thus, the heuristic directives are separately stored from the program and are all evaluated whenever a choice is made.

As an example, the following heuristic directive

```
#heuristic c(X) :
    x(X), not b(X), S = #sum{Y : b(Y)},
    W = S * 10 + X. [W@1]
is translated to the following Prolog query:
x(x), \+ b(x),
aggregate_all(sum(Y), b(Y), _0), S is _0,
WEIGHT is S * 10 + X, LEVEL is 1, \+ c(X).
```

The Prolog predicate `aggregate_all(+Template, : Goal, -Result)` aggregates bindings in `Goal` according to `Template`. Possible template values comprise the aggregate functions, `count`, `sum(X)`, `max(X)`, and `min(X)`. The variable in `sum(X)`, `max(X)`, and `min(X)` corresponds to the variable serving as aggregate term and is instantiated by querying `Goal` which contains this variable. The result is bound to an anonymous variable (`_0` in our example) and exploited in the aggregates' guards. Any negated atom is preceded by the operator `\+`, equivalent to `not` for our purposes. Finally, the negated head atom of the heuristic directive is added to the query to exclude already assigned head atoms.

During problem-solving, Alpha synchronizes the assignments with the database of the Prolog system. Every atom assigned as true by Alpha is inserted in the Prolog database. If such atoms are removed from the assignment, the corresponding facts are retracted from the Prolog database. Atoms assigned to false or must-be-true by Alpha are not considered since the heuristic directives are evaluated on atoms assigned to true.

The current implementation of Alpha sources and binaries which implement query-driven heuristics can be found on https://github.com/tilmanni/Alpha/tree/domspec_heuristics_extended_prolog.

6. Evaluation

We tested our approach to declarative domain-specific query-driven heuristics by creating heuristics for two example domains and applying the extended Alpha system. The two concrete domains under investigation were the Partner Units Problem (PUP) and the Balanced Sum Problem (BSP) introduced in Section 3.

These two problems are abstracted variants of typical configuration (sub)problems experienced in more than 25 years of applying AI technology in the automated configuration of electronic systems [3]. To put ASP systems under stress, we used problem encodings and instances of

varying sizes, where the larger instances were challenging to ground and/or to solve. More precisely, traditional grounders excessively consumed space or time when grounding these instances, and/or solving was infeasible without domain-specific heuristics.

6.1. Experimental setup

Encodings (including heuristics) and instances used for our experiments are available online.²

Alpha was used without justification analysis [12] (command-line argument `-dj`) and without support for negative integers in aggregates (`-dni`). Apart from that, Alpha was used in its default configuration. The JVM running Alpha was called with command-line parameters `-Xms1G -Xmx24G`, thus initially allocating 1 GiB for Java's heap and setting the maximum heap size to 24 GiB. The Prolog interpreter swi-prolog³ [13] version 9.0.4 was integrated with Alpha via jpl.⁴ For comparison, clingo⁵ [14] was used in version 5.6.2.

Each of the machines used to run the experiments ran Ubuntu 22.04.2 LTS Linux and was equipped with two Intel® Xeon® E5-2650 v4 @ 2.20GHz CPUs with 12 cores. Hyperthreading was disabled and the maximum CPU frequency was set to 2.90GHz. Scheduling of benchmarks was done with slurm⁶ version 21.08.5. runsolver⁷ v3.4.1 was used to limit time consumption to 10 minutes per instance and memory to 32 GiB. Care was taken to avoid side effects between CPUs, e.g., by requesting exclusive access to an entire machine for each benchmark.

All solvers were configured to search for the first answer set of each problem instance. Finding one or only a few solutions is often sufficient in industrial use cases since solving large instances can be challenging [3]. Therefore, the domain-specific heuristics used in the experiments are designed to help the solver find one answer set that is “good enough”, even though it may not be optimal.

6.2. Case Study 1: The Partner Units Problem (PUP)

The Partner Units Problem (PUP) [15] is an abstracted version of industrial configuration problems. In particular, PUP deals with configuring parts of railway safety systems. This problem is a benchmark problem for ASP systems since its challenges for grounding and solving.

²https://github.com/tilmanni/Alpha/tree/domspec_heuristics_extended_prolog/Evaluation

³<https://www.swi-prolog.org/>

⁴<https://jpl7.org/>

⁵<https://potassco.org/clingo/>

⁶<https://slurm.schedmd.com/>

⁷<https://github.com/utpalbora/runsolver>

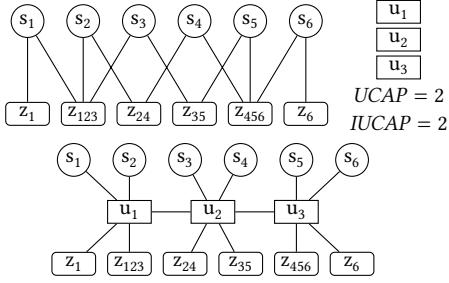


Figure 2: Sample PUP instance and one of its solutions [6]

Definition 6 (PUP). The input to the (PUP) is given by a set of units U and a bipartite graph $G = (S, Z, E)$ (also called the input graph), where S is a set of sensors, Z is a set of zones, and E is a relation between S and Z .

The task is to find a partition of vertices $v \in S \cup Z$ into bags $u_i \in U$ such that for each bag the following requirements hold: (1) the bag contains at most $UCAP$ vertices from S and at most $UCAP$ vertices from Z ; and (2) the bag has at most $IUCAP$ adjacent bags, where the bags u_1 and u_2 are adjacent whenever $v_i \in u_1$ and $v_j \in u_2$ for some $(v_i, v_j) \in E$.

We say a unit u_i is connected to a sensor/zone iff the sensor/zone is in u_i . Two units are connected iff they are adjacent. Connections correspond to physical connections in an assembled configuration.

Figure 2 shows an example of a PUP instance. The bipartite graph comprises six sensors and six zones. Each of the three units can be adjacent to at most two other units, and each unit can contain at most two sensors and two zones. Connections of sensors, zones, and units that satisfy all PUP requirements are presented in Figure 2.

Encodings and instances. To show the application and effectiveness of query-driven heuristics, we focus on the PUP instances employed in the ASP competition [5]. Domain-specific heuristics allow exploiting knowledge about properties of classes of problem instances. We concentrate on the double and double-variant classes of PUP instances. For these instances, domain-specific heuristics were formulated.

Figure 3 shows the basic structure of the double instances. There are two rows of rooms connected by doors. Each room corresponds to a zone, and each door represents a sensor. For each room and the doors of this room, there is an edge in the bipartite graph G , i.e., the zone and its doors are connected through an edge. The double instances' sizes vary depending on the number of columns of rooms. The structure depicted in Figure 3 shows three columns of rooms. The bipartite graphs G of the double-variant instances comprise the nodes and edges of the double instances. However, each dotted rectangle represents an additional zone (i.e., the dotted rectangle clusters rooms). Each door (i.e., a sensor) next

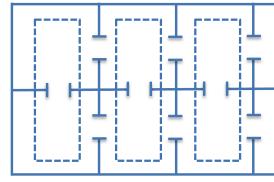


Figure 3: Double and double-variant instances

to a dotted rectangle (i.e., a zone) is connected by an edge in G . Note that there is no edge between a door surrounded by a rectangle and the zone corresponding to this rectangle.

Heuristics. The double PUP instances can be solved efficiently without backtracking by formulating the following heuristic directives, which employ dynamic aggregates.

```
#heuristic assigned_sensor_unit(S, U) :
  assignable_sensor_unit(S, U),
  not sensor_blocked_on_unit(S, U),
  Deg_sensor_dyn =
    #count{Z : zone2sensor(Z, S),
           assigned_zone_unit(Z, _)},
  Forbidden_placement_total =
    #max{N : num_forbidden_places_of_sensors(S, N)},
  Assigned_sensors_unit =
    #count{SN : assigned_sensor_unit(SN, U)},
  Direct_con_zones =
    #count{Z : assigned_zone_unit(Z, U),
           zone2sensor(Z, S)},
  W =
    Deg_sensor_dyn * 10000 +
    Forbidden_placement_total * 1000 +
    Assigned_sensors_unit * 100 +
    Direct_con_zones * 10.[W@0]
```

The atom `assignable_sensor_unit(S, U)` is true if a sensor is ready to be assigned, i.e., if a sensor is connected to a zone in the input graph and this zone is connected to a unit. The atom `sensor_blocked_on_unit(S, U)` is true if sensor S cannot be connected to unit U . The variable `Deg_sensor_dyn` is assigned to the number of zones connected to sensor S in the input graph and which are already assigned to a unit. The atom `zone2sensor(Z, S)` encodes the edges of the input graph (i.e., connections between zones and sensors). Values of the variable `Deg_sensor_dyn` express the number of constraints put on placing sensor S . We prefer connecting sensors to units with a higher number of constraints. The atom `num_forbidden_places_of_sensors(S, N)` represents the number of connections to units that are not possible for S for a given set of connections between sensors, zones, and units (e.g., the configuration in a specific

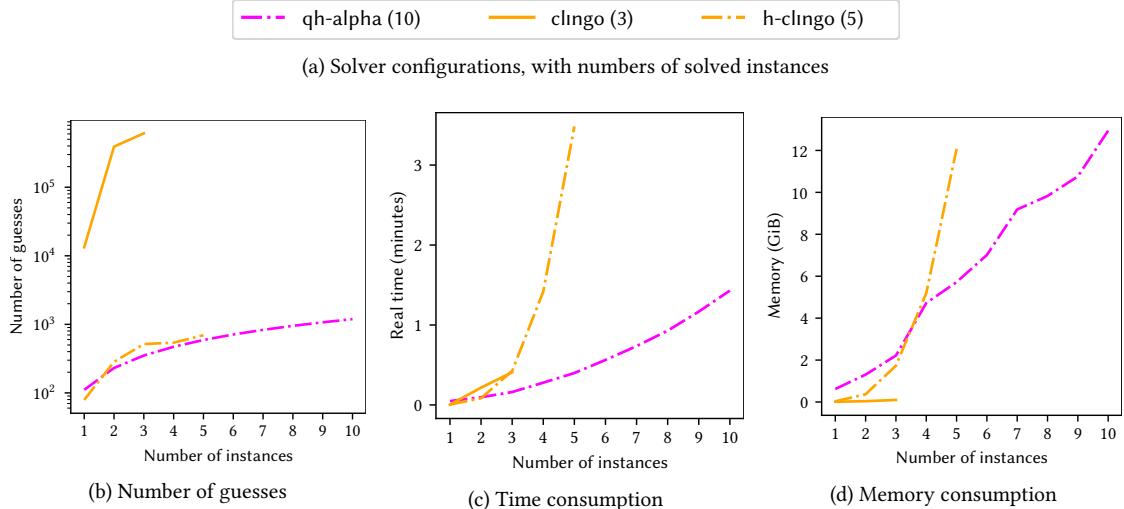


Figure 4: Resource consumption for solving each PUP Double instance

partial assignment). Rules compute different numbers depending on the connections. We prefer connecting sensors to units with a higher number of forbidden places (i.e., connections). The variable `Assigned_sensors_unit` counts the number of sensors connected to unit U. The variable `Direct_con_zones` counts the number of zones that are connected to unit U and which are connected to sensor S in the input graph. All these numbers are added with different weights resulting in the final weight of the heuristic directive expressing the priority to connect S and U. The design of the heuristic directive follows the principle of preferring assignments of connections that are most constrained in the spirit of heuristics of heuristics for constraint satisfaction problems (CSPs) such as “fail-first” or “degree” [16].

The second heuristic for assigning zones to units in double PUP instances can be formulated shorter than the presented one. We prefer assignments of zones to units U, where the number of connected zones to U is high, and the number of possible connections for sensors to U and its adjacent units is large.

Results. Figure 4 shows performance data for experiments with the double PUP instances. Cactus plots were created in the usual way. In Figure 4c, the x-axis gives the number of instances solved within real (i.e., wall-clock) time, given on the y-axis. Similarly, Figure 4b shows the number of guesses needed and Figure 4d shows the memory consumed to solve the instances. In all plots, data points are sorted by y-values. Figure 4a contains a legend with all solver configurations. The number of instances solved by each system is shown next to its name (in parentheses).

One curve was drawn for each solver configuration: Alpha with query-driven evaluation of domain-specific heuristics (qh-alpha), and clingo with (h-clingo) and without domain-specific heuristics.

Figure 5 shows the results for the double-variant instances in exactly the same way.

Alpha was used with encodings and heuristics designed to achieve a good performance as described above. clingo was used with the “new” encoding from the Fifth ASP competition⁸ [17]. h-clingo used the domain-specific heuristics devised in previous work [6]. Both systems used the same sets of problem instances, which consisted of 10 instances of the “double” class (with a number of units ranging between 20 and 200), and 6 instances of the “double-variants” class (with 30–180 units).

Substantial differences can be observed. The curves for qh-alpha reach farthest to the right, meaning that Alpha with query-driven heuristics solved the highest number of instances (all 10 double, 5 of 6 double-variants). clingo needed more time and thus solved fewer instances. Apparently, the domain-specific heuristics used with h-clingo were not useful for solving the double-variants instances.

6.3. Case Study 2: The Balanced Sum Problem (BSP)

The second evaluation case deals with the BSP. In configuring, sub-problems arise where quantities such as power consumption should be equally distributed.

⁸https://www.mat.unical.it/aspcomp2014/#Participants.2C_Encodings.2C_Instance_Sets

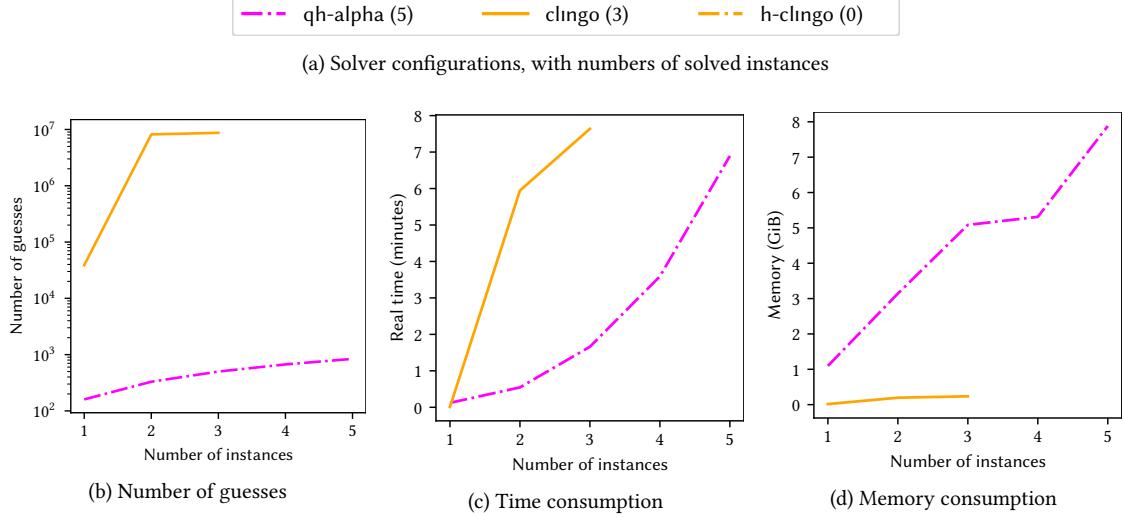


Figure 5: Resource consumption for solving each PUP DoubleV instance

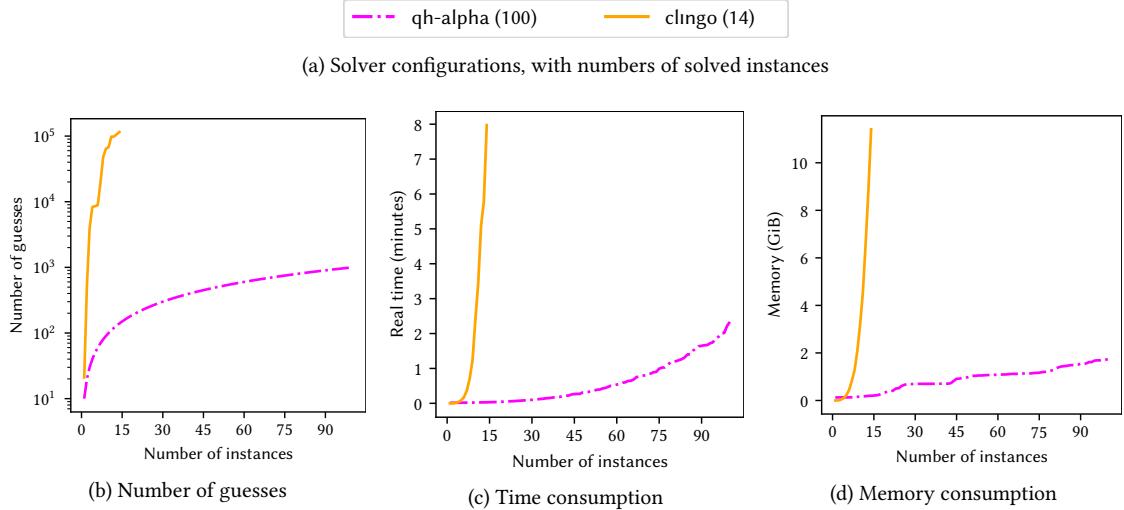


Figure 6: Resource consumption for solving each BSP instance

Encodings and instances. As the encoding of the problem, we use the ASP code introduced in Section 3. To strain the problem-solving, we increment the number of $x/1$ atoms and adapt the constant in the rules for $\text{sumB}/1$ and $\text{sumC}/1$.

Results. Results obtained for BSP are shown in Figure 6, which was generated in the same way as for PUP (cf. Section 6.2). clingo was used with the encoding presented in Section 3, while Alpha used an alternative representation of the sum constraints that the lazy-grounding system could evaluate more efficiently.

Since heuristics in the non-dynamic semantics are not known for this problem, clingo was only used without domain-specific heuristics. A hundred instances with $n \in \{10, 20, \dots, 990, 1000\}$ were used for the experiments.

In the BSP experiments, qh-alpha greatly outperformed clingo, showing the benefits of domain-specific heuristics evaluated in a query-driven way within a lazy-grounding ASP solver. While qh-alpha solved all 100 instances within at most 2.33 minutes per instance, clingo reaches the grounding bottleneck very quickly and is not able to solve instances larger than $n = 140$.

7. Conclusions and future work

Dynamic heuristics are an effective means for formulating domain-specific heuristics to speed up problem-solving. We have introduced dynamic aggregates, allowing us to reason about the current problem-solving state. E.g., we can reason about second-order properties of this state, such as the number of atoms with specific properties or summing quantities over sets of atoms or computing the maximum/minimum of such quantities. We have provided the prototypical implementation qh-alpha by integrating Prolog with the lazy-grounding system Alpha. This system was evaluated on two problems related to configuration, i.e., the double and double variant cases of the well-known Partner Units Problem, and the Balanced Sum Problem. However, dynamic aggregates are a general concept that knowledge engineers can apply to other problem domains. The evaluation shows that dynamic aggregates employed in domain-specific heuristics can considerably improve solving performance.

In future work, we plan to develop further the prototypical implementation to incorporate standard semantics of aggregate elements and terms. Additionally, exploring incorporating sign set semantics in query-driven heuristics would further enhance expressiveness. Efforts should also be made to utilize query-driven and regular domain-specific heuristics concurrently. Finally, we recommend evaluating the feasibility of applying the query-driven approach to specific aggregates in normal rules as an alternative to resource-intensive normalization.

Acknowledgments

The authors are grateful to the TU Wien for providing access to computational resources for running the experiments and to Tobias Geibinger for helping with using this infrastructure.

References

- [1] M. Gebser, R. Kaminski, B. Kaufmann, M. Lindauer, M. Ostrowski, J. Romero, T. Schaub, S. Thiele, P. Wanko, Potassco guide version 2.2.0, 2019. URL: <https://github.com/potassco/guide/releases/tag/v2.2.0>.
- [2] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, *Künstliche Intell.* 32 (2018) 165–176.
- [3] A. A. Falkner, G. Friedrich, A. Haselböck, G. Schenner, H. Schreiner, Twenty-five years of successful application of constraint technologies at Siemens, *AI Magazine* 37 (2016) 67–80.
- [4] M. Gebser, R. Kaminski, A. König, T. Schaub, Advances in gringo series 3, in: LPNMR, volume 6645 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 345–351.
- [5] M. Gebser, M. Maratea, F. Ricca, The seventh answer set programming competition: Design and results, *Theory Pract. Log. Program.* 20 (2020) 176–204.
- [6] R. Comploi-Taupe, G. Friedrich, K. Schekotihin, A. Weinzierl, Domain-specific heuristics in answer set programming: A declarative non-monotonic approach, *J. Artif. Intell. Res.* 76 (2023) 59–114.
- [7] A. Weinzierl, Blending lazy-grounding and CDNL search for answer-set solving, in: LPNMR, volume 10377 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 191–204.
- [8] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, T. Schaub, ASP-Core-2 input language format, *Theory Pract. Log. Program.* 20 (2020) 294–309.
- [9] L. Liu, E. Pontelli, T. C. Son, M. Truszczynski, Logic programs with abstract constraint atoms: The role of computations, in: ICLP, volume 4670 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 286–301.
- [10] C. Lefèvre, C. Béatrix, I. Stéphan, L. Garcia, AS-PeRiX, a first-order forward chaining approach for answer set computing, *Theory Pract. Log. Program.* 17 (2017) 266–310.
- [11] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, S. Malik, Chaff: Engineering an efficient SAT solver, in: DAC, ACM, 2001, pp. 530–535.
- [12] B. Bogaerts, A. Weinzierl, Exploiting justifications for lazy grounding of answer set programs, in: IJCAI, ijcai.org, 2018, pp. 1737–1745.
- [13] J. Wielemaker, T. Schrijvers, M. Triska, T. Lager, Swi-prolog, *Theory Pract. Log. Program.* 12 (2012) 67–96. doi:10.1017/S1471068411000494.
- [14] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, *Theory Pract. Log. Program.* 19 (2019) 27–82.
- [15] E. C. Teppan, Solving the partner units configuration problem with heuristic constraint answer set programming, in: Configuration Workshop, 2016, pp. 61–68.
- [16] S. J. Russell, P. Norvig, *Artificial Intelligence – A Modern Approach*, Fourth Edition, Pearson Education, 2022.
- [17] F. Calimeri, M. Gebser, M. Maratea, F. Ricca, Design and results of the fifth answer set programming competition, *Artif. Intell.* 231 (2016) 151–181. doi:10.1016/j.artint.2015.09.008.

Towards a formalization of configuration problems for ASP-based reasoning: Preliminary report

Nicolas Röhling^{1,2,*}, Torsten Schaub^{1,2} and Tobias Stolzmann^{1,2}

¹University of Potsdam, Germany

²Potassco Solutions, Germany

Abstract

We develop a principled approach to configuration that targets Answer Set Programming by integrating established concepts in a uniform setting. We begin by defining an abstract specification of configuration problems, drawing on concepts from the literature. We define both, user requirements and configuration solutions, as (partial) instantiations of a configuration model, and require the latter to be an extension of the former. The core of our configuration models comprise a partonomic structure which is adorned with constraints over atomic and aggregated attributes. Driven by this principled approach, we then develop a domain-independent ASP encoding for configuration.

Keywords

Answer Set Programming, Configuration, Encoding

1. Introduction

Configuration has been a central topic in AI since several decades [1, 2, 3]. Early on already, non-monotonic formalisms emerged as a promising alternative for modeling configuration problems [4]. Nowadays, this role is filled by Answer Set Programming (ASP) [5], a non-monotonic problem solving paradigm, combining an easy, rule-based modeling language with high performance solving capacities [6, 7]. Over the years, this has led to several applications of ASP to configuration problems, among them [8, 9, 10] and notably the ASP-based configuration systems WeCoTin [11] and VariSales [12].

Our objective is to develop a principled ASP-oriented approach to configuration, which integrates established concepts from configuration in a uniform setting. However, while ASP usually strives for generality, aiming at problem encodings covering the greatest possible class of problems, many approaches to configuration appear to be more down-to-earth, targeting more specific classes of configuration problems. Hence, as an intermediate step, we begin by defining an abstract specification of configuration problems, drawing on concepts borrowed from [13, 2, 14]. More precisely, we describe configuration problems in terms of a configuration model, user requirements and resulting configuration solutions [15]. Both user requirements and solutions are defined as (partial) instantiations of the configuration model [13], where

the latter is required to be an extension of the former. The core of our configuration models comprise a partonomic structure which is adorned with constraints over atomic and aggregated attributes.

Driven by this principled approach, we then develop a domain-independent ASP encoding for configuration.

The paper is structured as follows. In section 2 we formally define a configuration problem and its solutions. Section 3 discusses how constraints and aggregation of values are handled. In section 4 we present the ASP fact format and encoding and show how to solve configuration problems using the ASP solver *clingo*. Section 5 gives an overview of related work and section 6 concludes the paper.

2. Configuration problem and solutions

We represent configuration problems as (configuration) models along with one of their (partial) instantiations. Formally, both are expressed in terms of (directed) multigraphs;¹ the model's graph delineates the ones capturing partial instantiations. User requirements and solutions are both represented by instantiations.

A *configuration problem* is a pair (M, I) . A simple example is given in Figure 1. The (configuration) model M is a tuple $(T, P, s_p, t_p, D, V, E, C, de, at, co)$, where

1. (T, P, s_p, t_p) is a multigraph, where
 - a) T is a set of *types*,
 - b) $P = P_P \cup P_C$ is a partition of *ports*, where
 - i. P_P is a set of *partonomic ports*,
 - ii. P_C is a set of *connection ports*,

¹A *multigraph* is a graph admitting more than one arc between two vertices; for this, we use edges with own identity.

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

✉ nruehling@uni-potsdam.de (N. Röhling)
ID 0000-0001-5157-6788 (N. Röhling); 0000-0002-7456-041X
(T. Schaub); 0000-0002-1436-0715 (T. Stolzmann)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

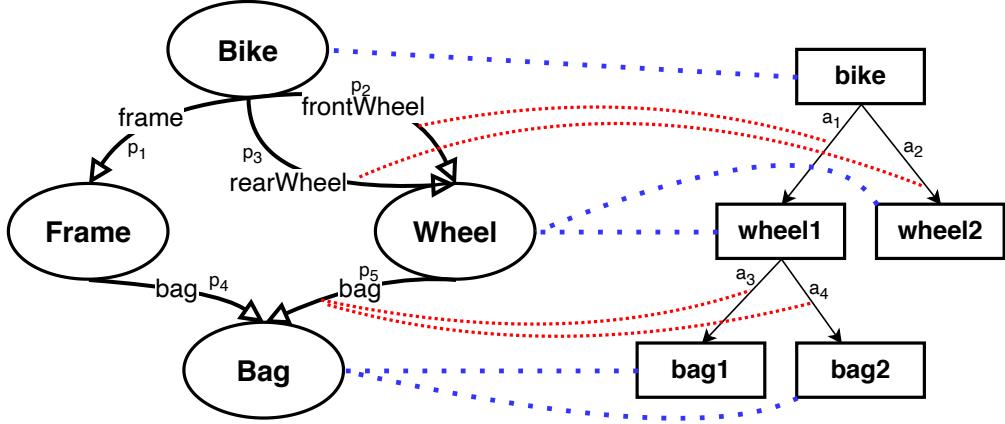


Figure 1: Example of a simple configuration problem.

- c) $s_p : P \rightarrow T$ assigns a port its *source type*,
- d) $t_p : P \rightarrow T$ assigns a port its *target type*, and
- e) (T, P_P, s_p, t_p) is a rooted acyclic graph,
- 2. $D = D_P \cup D_A$ is a partition of *descriptors*, where
 - a) D_P is a set of *port descriptors*,
 - b) D_A is a set of *attribute descriptors*,
- 3. V is a set of *values*,
- 4. E is a set of *evaluators*,
- 5. C is a set of *table constraints*,
- 6. $de : P \rightarrow D_P$ assigns a port its port descriptor, such that
 - if $s_p(p) = s_p(p')$ and $de(p) = de(p')$ then $p = p'$ for all $p, p' \in P$,
- 7. $at : T \rightarrow 2^{D_A \times E}$ assigns a type its set of attribute descriptors and evaluators, such that
 - for any type $t \in T$, if $(d, e), (d, e') \in at(t)$ then $e = e'$, and
- 8. $co : T \rightarrow 2^C$ assigns a type its set of constraints.

We often refer to (T, P, s_p, t_p) as the *model graph*, and to (T, P_P, s_p, t_p) as the *partonomy* (graph); its root represents the configured object.

An example of a model graph is given on the left in Figure 1. It consists of four types, $T = \{\text{Bike}, \text{Frame}, \text{Wheel}, \text{Bag}\}$, linked by five partonomic ports, viz. $P_P = \{p_1, p_2, p_3, p_4, p_5\}$ with source $s_p(p_1) = s_p(p_2) = s_p(p_3) = \text{Bike}$, $s_p(p_4) = \text{Frame}$, $s_p(p_5) = \text{Wheel}$ and target $t_p(p_1) = \text{Frame}$, $t_p(p_2) = t_p(p_3) = \text{Wheel}$ and $t_p(p_4) = t_p(p_5) = \text{Bag}$. The corresponding descriptors are $de(p_1) = \text{frame}$, $de(p_2) = \text{frontWheel}$, $de(p_3) = \text{rearWheel}$ and $de(p_4) = de(p_5) = \text{bag}$. Note that two ports can have the same descriptor as long as their source type is different. In usual graph terminology, this amounts to two edges $(\text{Bike}, \text{Wheel})$ labeled with *frontWheel* and *rearWheel*,

respectively. A third edge $(\text{Bike}, \text{Frame})$ labeled with *frame* and a fourth and fifth edge $(\text{Frame}, \text{Bag})$, resp. $(\text{Wheel}, \text{Bag})$, both labeled with *bag*. The other components of a configuration model are detailed in Section 3.

An instantiation I of M is a tuple $(O, A, s_a, t_a, m_O, m_A, X, v)$, where

1. (O, A, s_a, t_a) is a multigraph, where
 - a) O is a set of *objects*,
 - b) A is a set of *associations*,
 - c) $s_a : A \rightarrow O$ assigns an association its *source object*,
 - d) $t_a : A \rightarrow O$ assigns an association its *target object*,
2. $m_O : O \rightarrow T$ maps objects to types,
3. $m_A : A \rightarrow P$ maps associations to ports, such that for all $a \in A$
 - a) $m_O(s_a(a)) = s_p(m_A(a))$, and
 - b) $m_O(t_a(a)) = t_p(m_A(a))$,
4. $X = \{(o, d) \mid o \in O, (d, e) \in at(m_O(o)) \text{ for some } e \in E\}$ is a set of *attribute variables*, and
5. $v : X \rightarrow V$ maps attribute variables to values.

For simplicity, we sometimes drop the subscripts of m_O and m_A and simply write m , when clear from the type of argument.

An example instantiation of the configuration model in Figure 1 is given on its right. It includes objects $O = \{\text{bike}, \text{wheel1}, \text{wheel2}, \text{bag1}, \text{bag2}\}$ whose relationships are fixed via the associations $A = \{a_1, a_2, a_3, a_4\}$ with source $s_a(a_1) = s_a(a_2) = \text{bike}$ and $s_a(a_3) = s_a(a_4) = \text{wheel1}$, and target $t_a(a_1) = \text{wheel1}$, $t_a(a_2) = \text{wheel2}$, $t_a(a_3) = \text{bag1}$, and $t_a(a_4) = \text{bag2}$. The actual instantiation of the configuration model is warranted by functions m_O and m_A . The object mappings are

$m_O(bike) = Bike$, $m_O(wheel1) = m_O(wheel2) = Wheel$, and $m_O(bag1) = m_O(bag2) = Bag$. The association mappings are $m_A(a_1) = p_2$ and $m_A(a_2) = p_3$, and $m_A(a_3) = m_A(a_4) = p_5$. There are no corresponding objects and associations for type *Frame* and partonomic port *frame*, respectively. This shows that partial instantiations are fully admissible.

The other components of instantiations are detailed in Section 3.

Finally, a *valid instantiation* I of M satisfies the following conditions:

1. All constraints in $co(m(o))$ are satisfied for all $o \in O$, and
2. the subgraph (O, A_P, s_a, t_a) , where $A_P = \{a \in A \mid m_A(a) \in P_P\}$ is the set of all *partonomic associations*, is a tree with root $r \in O$ such that $m_O(r)$ is the (partonomic) root of (T, P_P, s_p, t_p) .

The satisfaction of constraints is detailed in Section 3. The first condition ensures consistency of the instantiation while the second guarantees that it is indeed a configuration of the object in focus where every non-root object is a part of exactly one other object.

For comparing instantiations in terms of partiality, we view all components as sets (i.e., functions as relations) and compare them with set inclusion. Accordingly, we say that an instantiation I' is an *extension* of another I , written $I \prec I'$, if all components of I are subsets of the ones of I' . In this way, we may pose a configuration problem (M, U) as a configuration model M along with *user requirements* U expressed as a (partial) instantiation of M . We define the set of *solutions* to (M, U) as

$$S(M, U) = \{I \mid I \text{ is a valid instantiation of } M \text{ and } U \preceq I\}.$$

This assures the user requirements are included in any solution; invalid user requirements or ones which cannot be extended to a valid instantiation may lack solutions.

A *minimal solution* of some user requirements U is a valid extension $U \prec I$ such that there is no valid instantiation I' with $U \prec I' \prec I$.

One might wonder why a rigorous specification of configuration problems as shown above is necessary. Such a specification allows us to show properties of our formalism. For example, there is a simple proof that the solution space behaves monotonically for a fixed model.

Proposition 1. *Let M be a fixed configuration model. Then for any user requirements U and U' it holds that $U \prec U'$ implies $S(M, U') \subseteq S(M, U)$.*

Proof. Take any $I \in S(M, U')$. Per definition I is valid and $U' \prec I$, that is, all components of U' are subsets of I . We also have $U \prec U'$ so all components of U are subsets

of U' . That means all components of U are also subsets of I and thus $U \prec I$. It follows that $I \in S(M, U)$. \square

Note that in general this does not hold for minimal solutions.

3. Constraint handling and aggregation

The objects in a valid instantiation must satisfy all associated constraints in the underlying configuration model. Apart from the structural constraints imposed by the model graph, additional constraints can be imposed on objects (in the instantiation) via their types.

As an example, consider Figure 2 showing the model graph from Figure 1 but with attributes and constraints. The model is extended by a set of constraints $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$, assigned to their corresponding types, viz. $co(Bike) = \{c_1, c_2, c_3\}$, $co(Wheel) = \{c_4, c_5\}$, and $co(Bag) = \{c_6\}$.

The attributes are assigned to their types as follows

$$\begin{aligned} at(Bike) &= \{(maxWeight, \top), (minStowage, \top), \\ &\quad (totalWeight, e_1), (stowage, e_2)\}, \\ at(Wheel) &= \{(size, \top), (weight, \top)\}, \text{ and} \\ at(Bag) &= \{(volume, \top), (weight, \top)\}. \end{aligned}$$

Here $\top, e_1, e_2 \in E$ are evaluators whose function is explained later in this section. While constraint c_1 guarantees that the front and rear wheel of a bicycle have the same size, constraints c_2 and c_3 assure that the values of the total weight and stowage of the bike lie within some (possibly user-requested) range. Constraints c_4 and c_5 specify possible combinations of the attributes of the wheels and bags. Lastly, constraint c_6 expresses that only small bags can be attached to a wheel.

We represent constraints in their canonical form as tables. For illustration of how table constraints work, consider the instantiation in Figure 3 and constraint $c_1 \in co(m(bike))$ from Figure 2. This constraint is expressed as an equality but can easily be rewritten as a table containing all combinations which satisfy the given relation. The constraint describes compatible values of the attribute *size* of *Wheel* at paths *frontWheel* and *rearWheel* of *Bike*.

To make this relation precise, we rely on *path expressions* leading from the type at hand to the attributes in focus. In our example, they are given in the header of the table constraint. Notably, given that this structure is mirrored in corresponding instantiations, the path expressions also allow us to access the values of these attributes from each object of the type at hand.

More precisely, a *path expression* is a finite sequence of descriptors. We distinguish path expressions only including port descriptors in D_P , and the ones consisting

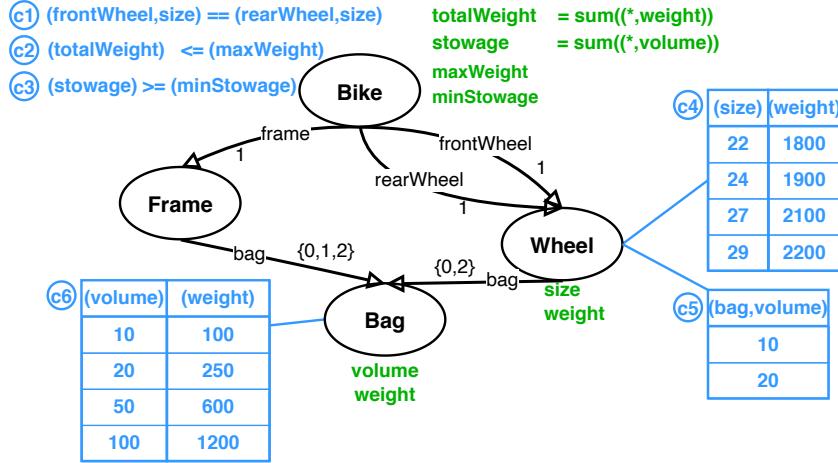


Figure 2: Model of bike example with constraints.

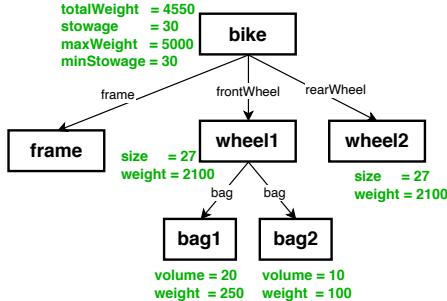


Figure 3: Instantiation of bike example.

of a sequence of port descriptors followed by a single attribute descriptor from D_A . We refer to them as port and attribute path expressions, respectively. Attribute path expressions are used as attributes in table constraints, as with $(frontWheel, size)$ or $(weight)$ in Figure 2.

With this, a *table constraint* is a pair $((\vec{d}_1, \dots, \vec{d}_n), R)$ where each \vec{d}_i is an attribute path expression for $1 \leq i \leq n$ and $R \subseteq V^n$ is an n -ary relation over V . We use path expressions to select objects as well as attribute variables. For an object $o \in O$ and $n \geq 0$, we define

$$sel_o(\epsilon) = \{o\} \text{ for the empty path sequence } \epsilon. \quad (1)$$

$$sel_o((d_1, \dots, d_n, d_{n+1})) = \{o' \in O \mid a \in A, t_a(a) = o', s_a(a) \in sel_o((d_1, \dots, d_n)), de(m(a)) = d_{n+1}\} \text{ if } d_{n+1} \in D_P \text{ and } d_i \in D_P \text{ for } 1 \leq i \leq n. \quad (2)$$

$$sel_o((d_1, \dots, d_n, d_{n+1})) = \{(o', d_{n+1}) \in X \mid o' \in sel_o((d_1, \dots, d_n))\} \text{ if } d_{n+1} \in D_A \text{ and } d_i \in D_P \text{ for } 1 \leq i \leq n. \quad (3)$$

In our example, for constraint $c_1 \in co(m(bike))$ we get

$$sel_{bike}((frontWheel)) = \{wheel1\} \quad (4)$$

$$sel_{bike}((rearWheel)) = \{wheel2\} \quad (5)$$

$$sel_{bike}((frontWheel, size)) = \{(wheel1, size)\} \quad (6)$$

$$sel_{bike}((rearWheel, size)) = \{(wheel2, size)\}. \quad (7)$$

While (6) and (7) give attribute path expressions, (4) and (5) give port path expressions.

Given an object $o \in O$ and a valuation v , o satisfies a table constraint $((\vec{d}_1, \dots, \vec{d}_n), R)$ in $co(m(o))$, if $(v(x_1), \dots, v(x_n)) \in R$ for every $(x_1, \dots, x_n) \in sel_o(\vec{d}_1) \times \dots \times sel_o(\vec{d}_n)$.

For the type *bike* in our example in Figure 3, we continue the illustration of constraint $c_1 \in co(m(bike))$ given by

$$(((frontWheel, size), (rearWheel, size)), \{(22, 22), (24, 24), (27, 27), (29, 29)\})$$

When using the two attribute path expressions to select attribute variables, we look at the cross product of (6) and (7):

$$\{((wheel1, size), (wheel2, size))\} \quad (8)$$

Applying the valuation v from Figure 3, namely,

$$v = \{(wheel1, size) \mapsto 27, (wheel2, size) \mapsto 27, \dots\}$$

to the cross product obtained in (8) yields tuple $(27, 27)$ which belongs to the binary relation of $c_1 \in co(m(bike))$. In this way, we can check satisfaction of all other constraints of the instantiation in Figure 3 which are $co(m(bike)) = \{c_1, c_2, c_3\}$, $co(m(wheel1)) = co(m(wheel2)) = \{c_4, c_5\}$, and $co(m(bag1)) = co(m(bag2)) = \{c_6\}$. Due to space limitations we do not work this out in detail but by comparing Figures 2 and 3 it is easy to see that all objects in our example instantiation satisfy the constraints imposed by their underlying types. Together with the fact that the instantiation in Figure 3 is a tree with root *bike* and $m_O(bike) = Bike$ constitutes the partonomic root of the model graph, we may conclude that our example is a valid instantiation of our configuration model.

The constraint illustrated above imposed a relation on attributes with atomic values. In addition, we want to account for attributes taking aggregated values. For example, the weight of a wheel is (usually) explicitly given, while that of an entire bike must be calculated from the weights of its components. Also, we can use calculated attributes for enforcing port multiplicities, as we show below. To this end, we allow for attributes whose value is either assigned or calculated via aggregate functions, like addition or maximum. We address this via the evaluators in E along with a refinement of the valuation function v . As above, we rely on path expressions for selecting the values subject to aggregation.

Accordingly, an evaluator $e \in E$ is either \top , indicating that an atomic value is assigned, or a pair $((\vec{d}_1, \dots, \vec{d}_n), f)$ where each \vec{d}_i is a path expression for $1 \leq i \leq m$ and f is an aggregate function. Aggregate functions are defined on sets and yield an element from V . When the input is the empty set, this is their neutral element, e.g., 0 for the function *sum*.

Given $o \in O$, we define for $(d, e) \in at(m_O(o))$

$$v((o, d)) = \begin{cases} v \in V & \text{if } e = \top \\ f \left(\bigcup_{i=1}^n sel_o(\vec{d}_i) \right) & \text{if } e = ((\vec{d}_i)_{i=1}^n, f) \end{cases}$$

This function combines the assignment of attributes to atomic and calculated values.

For simplicity, we often write $d = f(\vec{d}_1, \dots, \vec{d}_n)$ whenever $(d, ((\vec{d}_1, \dots, \vec{d}_n), f)) \in at(t)$ for some type $t \in T$. For example, in Figure 2 consider the attribute calculating the total weight of a bike indicated by $totalWeight = sum(*, weight)$. In the above notation, this corresponds to attribute $(totalWeight, e_1) \in at(Bike)$, with evaluator $e_1 = ((*, weight), sum)$. The expression $(*, weight)$ is syntactic sugar for all attribute path expressions pointing to an attribute *weight*, here expanding to the sequence

$((frontWheel, weight), (rearWheel, weight), (frontWheel, bag, weight), (rearWheel, bag, weight), (frame, bag, weight))$.

Accordingly, the value of the calculated attribute *totalWeight* of object *bike* is 4550, the sum of two individual wheel weights 2100 and individual bag weights of 250 and 100, as shown in the instantiation in Figure 3.

The above concepts also allow us to account for port multiplicities. This can be done by using a calculated attribute constrained by all legitimate multiplicities. As an example, consider the model in Figure 2 but with the type *Wheel* extended by an attribute $\#bags$ along with the evaluator $((bag), count)$ and the table constraint $((\#bags), ((0), (2)))$; it expresses that a wheel must have exactly 0 or 2 bags. Instead of writing out the constraint and auxiliary attribute, we often denote this by just adding “ $\{0, 2\}$ ” to the correponding arrow. Note that unlike above, the aggregator relies on a port path expression, yielding the bags *bag1*, *bag2* for *wheel1* and no objects for *wheel2*. Accordingly, the value of attribute $\#bags$ of *wheel1* (resp. *wheel2*) is 2 (resp. 0). This is among the admissible values of the constraint imposed by *Wheel*.

4. An ASP-based solution to configuration problems

For brevity, we refrain from giving an introduction to ASP. Full details on the input language of *clingo* along with various examples can be found in the *Potassco User Guide* [16].

4.1. Configuration model fact format

```

1 type((bike;wheel;frame;bag)).
3 part(bike,wheel,frontwheel).
4 multiplicity(bike,wheel,frontwheel,1).
5 part(wheel,bag,bag).
6 multiplicity(wheel,bag,bag,(0;2)).
```

Listing 1: Facts representing parts of the model graph of the bike example from Figure 2

Listings 1-4 display a snippet of the encoding representing the bike example from Figure 2. A part of the model graph is encoded in Listing 1. Types are declared via a *type/1* atom where the argument is the name of the type. Parts are declared via a *part/3* atom with source and target type and port descriptor as arguments. The corresponding multiplicites are encoded via a *multiplicity/4* atom with the same structure as the *part/3* atom plus all possible multiplicities as fourth argument.

```

1 attr(wheel,size).
2 dom(wheel,size,(22;24;27;29)).
3 attr(wheel,weight).
4 dom(wheel,weight,(1800;1900;2100;2200)).

```

Listing 2: Facts representing the *Wheel* attributes of the bike example from Figure 2

Listing 2 contains the encoding of attributes *size* and *weight* of type *Wheel*. Atomic attributes are declared via an *attr/2* atom with type and attribute descriptor as arguments together with domain *dom/3* atoms. As before, the domain atoms have the same structure as *attr/2* atoms plus the possible values as third argument.

```

1 attr(bike,totalWeight,"sum").
2 path(bike,totalWeight,
3   ((weight,(frontWheel,())));
4   (weight,(rearWheel,())));
5   (weight,(bag,(frontWheel,())));
6   (weight,(bag,(rearWheel,())));
7   (weight,(bag,(frame,())))).

```

Listing 3: Facts representing a calculated attribute of the bike example from Figure 2

Listing 3 contains the encoding of the calculated attribute *totalWeight* from type *Bike*. Calculated attributes are declared via an *attr/3* atom. The structure is the same as *attr/2* atoms plus the aggregate function as third argument (currently *sum*, *count*, *min* and *max* are supported). The corresponding path expressions which are used to gather all values are declared via *path/3* atoms. The first two arguments have to be the same as the *attr/3* atoms and the third argument is a path expression. Path expressions follow a nested tuple structure in ASP with the first element of the sequence being the innermost. Consider for example the path expression (d_1, d_2, d_3) in the formalism. We express this in ASP as $(d_3, (d_2, (d_1, ())))$. While this is not easily readable for humans, it enables one to work dynamically with tuples of any size in ASP. In the future, we plan to implement an input and output language which is human-readable and leave the nested tuple structure for internal representation.

Constraints are declared via a *constraint/1* atom where the argument is a *constraint identifier*. In Listing 4 an example of a table constraint is shown. The identifier is a tuple consisting of the type the constraint is attached to and an index. The columns of the table are declared via *column/3* atoms containing the constraint identifier, the index of the column and the path expression. The actual entries are declared via *entry/3* atoms containing the constraint identifier, a tuple with the index of the column and row, and the value of the entry.

```

1 constraint((wheel,0)).
2 column((wheel,0),0,(size,())).
3 column((wheel,0),1,(weight,())).
4 entry((wheel,0),(0,0),22).
5 entry((wheel,0),(1,0),1800).
6 entry((wheel,0),(0,1),24).
7 entry((wheel,0),(1,1),1900).
8 entry((wheel,0),(0,2),27).
9 entry((wheel,0),(1,2),2100).
10 entry((wheel,0),(0,3),29).
11 entry((wheel,0),(1,3),2200).

```

Listing 4: Facts representing a table constraint of the bike example from Figure 2

4.2. General problem encoding

The ASP encoding of our formalization can be found in Listing 5. In Lines 1-6 it is checked that the configuration model graph is indeed acyclic and rooted. Further, the type of the root object is determined which is created in Line 8 (with the correct type as second argument). In Lines 10-12 objects for each part relation are generated while making sure that the indices of the objects are assigned incrementally. Satisfaction of the multiplicities of those part relations are assured in Lines 14-15. In Lines 17-19 values are assigned to attribute values according to their domain making sure that each object has exactly one value assigned for all its attributes. All possible port and attribute selectors are created in Lines 22-24. Using selectors, the correct values for aggregates are determined and assigned. In Lines 26-27, we show the encoding for one such aggregate function *sum*. Our full implementation which can be found under <https://github.com/potassco/configuration-encoding> also contains the aggregate functions *count*, *min* and *max*. Lastly, in Lines 29-42 table constraint satisfaction is checked for each object. First, all possible tuples of the cross product are created (encoded again as nested tuples). Then the tuples are unpacked step-by-step while traversing the columns of the constraint. Only if all tuples satisfy at least one complete row, the constraint is satisfied.

Due to space limitations, we are not showing the full implementation of our formalization. As mentioned above, we are only showing the aggregate function *sum*. Additionally, we left out connection ports and comparison constraints (e.g., $=$, \leq , etc.) in Listing 5. In our full encoding we also distinguish between *mandatory objects* (encoded by normal rules) and *optional objects* encoded by choice rules by which we hope to achieve a better performance. In addition to that, several examples and files to visualize configuration models and instantiations using *clingo* [17] can be found in the repository linked earlier.

```

1 partonomic_path(X,Y) :- part(X,Y,_).
2 partonomic_path(X,Z) :- partonomic_path(X,Y), partonomic_path(Y,Z).
3 :- partonomic_path(X,X).

5 root(T) :- type(T), not partonomic_path(_,T).
6 :- {root(T)} > 1.

8 object(((),T) :- root(T).

10 { object((D,(O,I)),T) : I = 0..Max-1 } :-  

11     object(O,S), part(S,T,D), Max = #max { N : multiplicity(S,T,D,N)}.
12 :- object((D,(O,I)),_), not object((D,(O,I-1)),_), I > 0.

14 :- part(S,T,D), object(O,S), not multiplicity(S,T,D,X),
15     X = #count { I : object((D,(O,I)),T) }.

17 { val((O,D),V) : dom(T,D,V) } :- object(O,T), attr(T,D).
18 :- attr(T,D), object(O,T), not val((O,D),_).
19 :- val(X,V1), val(X,V2), V1 < V2.

21 attr(T,D,"atomic") :- attr(T,D).
22 selector(O,((),O) :- object(O,_).
23 selector(O,(D,P),(D,(O',I))) :- selector(O,P,O'), object((D,(O',I)),_).
24 selector(O,(D,P),(O',D)) :- selector(O,P,O'), object(O',T), attr(T,D,_).

26 val((O,D),V) :- object(O,T), attr(T,D,"sum"),
27     V = #sum { V',X,P : path(T,D,P), val(X,V'), selector(O,P,X) }.

29 max_col_idx(C,N) :- constraint(C), N = #max{ Col : column(C,Col,...)}.
30 tuple((O,C),N,((),X)) :- object(O,T), max_col_idx((T,C),N),
31     selector(O,P,X), column((T,C),N,P).
32 tuple((O,C),N,(VT,X)) :- object(O,T), tuple((O,C),N+1,VT),
33     selector(O,P,X), column((T,C),N,P), N>=0.

35 sat_row((O,C),VT,(0,Row),VT') :-  

36     object(O,T), tuple((O,C),0,VT), VT = (VT',X),
37     val(X,V), entry((T,C),(0,Row),V).
38 sat_row((O,C),VT,(Col,Row),VT'') :-  

39     object(O,T), sat_row((O,C),VT,(Col-1,Row),VT'),
40     VT'' = (VT'',X), val(X,V), entry((T,C),(Col,Row),V).

42 :- tuple(C,0,VT), not sat_row(C,VT,_,()).

```

Listing 5: ASP encoding for solving configuration problems.

4.3. Instantiation fact format and obtaining solutions

On the instantiation level, there are two important atoms `object/2` and `val/2`. They represent objects and the valuations of attribute variables, respectively. The `object/2` atom takes as arguments the name of the object encoded as a nested tuple and its type. The nested tuple structure is similar as for path expressions above (see Section 4.1). The names are constructed from the partonomic port descriptors and indices. Take for example the atom

`object((bag,((frontWheel,(((),0)),1)),bag).` This corresponds to the second bag of the first (and only) wheel with descriptor *frontWheel* of the root object (which has type *Bike*). The root is always encoded as an empty tuple `()`. Note that this way of encoding objects directly assures that the set of partonomic associations is a tree as required for valid instantiations.

The `val/2` atom takes as first argument an attribute variable encoded as a tuple. The tuple contains the object name and the attribute descriptor. The second argument of the atom is the actual value of the variable. For example, we have the atom `val((((),minStowage),30)`

which expresses that attribute *minStowage* of the root object bike has value 30.

We can run the encoding together with the file of a model M to obtain one or multiple stable models. The stable models correspond to valid instantiations as defined in Section 2. This is easy to verify, as (table) constraints are encoded as integrity constraints in ASP, thus have to be satisfied in every stable model. Further, as mentioned above our object structure directly assures that the set of partonomic associations is a tree and that the root object has the type of the partonomic root of the model graph. We can also specify user requirements U by providing, e.g. another input file instantiation.lp. Every instantiation obtained in form of a stable model then extends the user requirements and is therefore a solution to (M, U) .

In Listing 6 we run our full encoding with the model from Figure 2. The user requirements are empty, i.e., omitted, and the solution we obtain corresponds to the one from Figure 3.

```
$ clingo encoding.lp examples/bike/model.lp
clingo version 5.6.2
Reading from encoding.lp ...
Solving...
Answer: 1
object((),bike)
object((frontWheel,(((),0)),wheel)
object((rearWheel,(((),0)),wheel)
object((frame,(((),0)),frame)
object((bag,((frontWheel,(((),0)),0)),bag)
object((bag,((frontWheel,(((),0)),1)),bag)
val((((),maxWeight),5000)
val((((),minStowage),30)
val(((frontWheel,(((),0)),size),27)
val(((rearWheel,(((),0)),size),27)
val(((frontWheel,(((),0)),weight),2100)
val(((rearWheel,(((),0)),weight),2100)
val(((bag,((frontWheel,(((),0)),0)),volume),
    ,20)
val(((bag,((frontWheel,(((),0)),0)),weight),
    ,250)
val(((bag,((frontWheel,(((),0)),1)),volume),
    ,10)
val(((bag,((frontWheel,(((),0)),1)),weight),
    ,100)
val((((),stowage),30)
val((((),totalWeight),4550)
SATISFIABLE
```

Listing 6: Running the bike example from Figure 2 in *clingo*

5. Related work

Our formalism borrows concepts from various other approaches in the literature. A general ontology of configuration has been introduced in [13]. Here, a configuration problem is divided into *configuration model knowledge*,

configuration solution knowledge and *requirements knowledge*. However, the paper argues that the latter can be expressed in terms of the other two. In the model knowledge there are *product specific classes* called types and a *configuration* of a product w.r.t. to a configuration model is defined as a set of instances of the types occurring in the model. These instances are called *individuals*. Constraints are specified inside the model and a correct configuration must satisfy these. However, the definition of constraints is left to an unspecified constraint language. A configuration also contains configuration specific relations called *properties*. Unlike our approach, [13] includes the concepts of taxonomy and inheritance.

Another formal approach to configuration in the context of constraint programming has been given by [2]. Here, a *structural configuration model* lays out the possible variations of the entity to be configured. This model contains types and attributes, as well as partonomic and connection ports. Types can be functional or technical and this restricts the possible kinds of (taxonomic) subtypes they are allowed to have. Technical types can only have concrete types as subtypes which can be seen as "complete" parts ready to be ordered from a catalog. A *configuration* can be obtained from a structural model by instantiating types. Instances inherit the attributes and ports from their type and all its supertypes. As to what regards constraints, three kinds are defined: compatibility, requirement and resource constraints.

Lastly, [14] follows a somewhat less formal, object-oriented approach at modelling configuration problems where concepts are directly defined in ASP. Again, there is a distinction between a *model* and an *instantiation*. The model contains a taxonomy of *classes* and a general association relation (with no distinction between part and connection relations). It is noteworthy, that associations in general have multiplicities in both directions. Further, attributes are limited to be over the domain of strings, integers or booleans. In an instantiation of a model, each object is defined through a global index. An "is-a" relation ties it to a class. Two objects are connected through an "associated" relation which has to correspond to an association relation from the model. *Attribute values* assign values to attributes of objects. Constraints are not specified directly but left open to general integrity constraints in ASP.

6. Discussion

We presented an approach to model and solve configuration problems and a corresponding encoding for ASP. A model and its instantiation are expressed through directed multigraphs where the former specifies the possible graphs of the latter. Similar to [2], the partonomy of the model graph has to be acyclic. We view this as

favorably for object generation.

Apart from the structural constraints imposed by the model graph, each type in the model may impose constraints on itself and its parts. For this we use table constraints as a canonical representation which specify possible combinations of attributes. In our view, table constraints are the most general form of constraints and other kinds can be expressed through them. Constraints attached to a type are checked independently for each object of that type. This guarantees that they are only applied in the correct context.

Attributes can be atomic, i.e., a value needs to be assigned, or calculated. For the latter we make use of aggregate functions. We also formalized the concepts of path expressions and selectors. The former can be composed of port or attribute descriptors and the latter return sets of attribute variables or objects. While port multiplicities are unbounded in general, we can use aggregate functions and port path expressions to restrict them.

In contrast to many other approaches, we cannot target specific objects in the instantiation with our constraints. For example, in Figure 3 it would not be possible to attach a constraint to the type *Wheel* targeting only *bag1*. This is because there is no selector starting at *wheel1* containing this bag only. Note that $sel_{wheel1}((bag))$ returns the set $\{bag1, bag2\}$ with both bags. This was done on purpose to evade symmetries. Our understanding is that if distinctions between objects are desirable, this information should be included in the model, e.g., by having separate ports as for the front and rear wheel.

In many scenarios it is desirable to configure multiple instances of the same type simultaneously. Since we require every configuration to be rooted, this might not appear possible within our approach. However, one could always add a new root type to the model, for example a *Fleet* of bikes.

A shortcoming of our formalization is that we require the attribute valuation function to be total, i.e., every instantiated attribute variable needs to have a value assigned. In user requirements, though, it might be desirable to leave certain attributes undefined.

Further, there are user requirements which cannot be expressed through an instantiation. For example, consider Figure 2 and a user who wants all bags to be of a certain color but does not care about the number of bags. This would require adding a new constraint to the model. Anyhow, we consider this to be more of a knowledge engineering problem as any such option should only be available if included in the model.

Compared to many other approaches that are tuned for practical applications, our approach is more abstract. This allows us to formally prove properties as we have demonstrated with the monotonicity of the solution space. Experience has shown that this is important when working with ASP. In the future, we intend to investigate

what other properties our formalism possesses.

Partly due to this abstractness, we decided to start with a simpler approach excluding a taxonomy and thus a form of inheritance. However, we view both these concepts as vital for efficiently modelling configuration problems and we plan to extend our formalism to contain them. This would probably require the following steps:

1. Introduce *taxonomic ports* representing a "super-type" relation. Following [2], our acyclicity condition would be extended to consider these ports as well.
2. Adapt the definition of a valid instantiation. Now the root object of the model graph does not necessarily represent the object to be configured but could be specialized to a subtype. There might not even be a partonomic root anymore. In general, it should be possible to start the configuration at any node of the model graph which could be expressed through user requirements and a dedicated "root object" in the instantiation.
3. Types would inherit attributes and other types of ports from their supertypes, i.e., constraint satisfaction would have to be checked not only for the type an object is mapped to but also for all its supertypes.
4. A more difficult question is how to treat attributes and (non-taxonomic) ports which appear more than once for a set of supertypes or if this should be prohibited.

Lastly, we note that in many examples a taxonomy only appears in form of "specializations" (such as concrete types and catalogs in [2]). This feature can be represented in our formalism by adding table constraints (see for example constraint *c4* in Figure 2 describing the possible wheels).

Further, we plan to further study connection ports by working out examples. They are part of the literature [2] and we view them as an important complement to partonomic ports since they convey additional information and allow to form cycles in the model graph. Consider the configuration of a computer network. Here, a configuration might have identical parts like switches which can be connected in numerous ways. We also intend to investigate symmetry conditions for connection ports which currently do not seem to be expressible within our formalization of constraints.

On the implementation level, our full encoding currently supports table and comparison constraints. In the literature other kinds such as requirement and incompatibility constraints often occur. We plan to study how these can be represented in our formalism and to add them to our implementation.

Acknowledgments

This work was partially funded by ZIM (Zentrales Innovationsprogramm Mittelstand) of the German Federal Ministry for Economic Affairs and Climate Action (grant number: KK5291302GR1). We thank Joachim Baumeister, Richard Comploi-Taube, Andreas Falkner, Konstantin Herud, Jochen Reutelshöfer and Gottfried Schenner for their valuable feedback.

References

- [1] R. Cunis, A. Günter, H. Strecker, Das PLAKON-Buch, volume 266 of *Informatik Fachberichte*, Springer-Verlag, 1991. doi:10.1007/978-3-662-06485-6.
- [2] U. Junker, Configuration, in: F. Rossi, P. van Beek, T. Walsh (Eds.), *Handbook of Constraint Programming*, Elsevier Science, 2006, pp. 837–873. doi:10.1016/s1574-6526(06)80028-3.
- [3] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen (Eds.), *Knowledge-Based Configuration: From Research to Business Cases*, Elsevier/Morgan Kaufmann, 2014. doi:10.1016/C2011-0-69705-4.
- [4] G. Brewka, T. Schaub, Zur Verwendung nichtmonotoner Inferenztechniken bei der Konfiguration, in: F. di Primio (Ed.), *Methoden der Künstlichen Intelligenz für Grafikanwendungen*, Addison-Wesley, 1995, pp. 45–60. In German.
- [5] V. Lifschitz, Answer Set Programming, Springer-Verlag, 2019. doi:10.1007/978-3-030-24658-7.
- [6] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, *Artificial Intelligence* 187–188 (2012) 52–89. doi:10.1016/j.artint.2012.04.001.
- [7] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, *Answer Set Solving in Practice*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers, 2012. doi:10.1007/978-3-031-01561-8.
- [8] M. Gebser, R. Kaminski, T. Schaub, aspcud: A Linux package configuration tool based on answer set programming, in: C. Drescher, I. Lynce, R. Treinen (Eds.), *Proceedings of the Second International Workshop on Logics for Component Configuration (LoCoCo’11)*, volume 65 of *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, 2011, pp. 12–25. doi:10.4204/eptcs.65.2.
- [9] A. Felfernig, A. Falkner, M. Atas, S. Erdeniz, C. Uran, P. Azzoni, ASP-based knowledge representations for IoT configuration scenarios, in: L. Zhang, A. Haag (Eds.), *Proceedings of the Nineteenth International Configuration Workshop (CONF’17)*, 2017, pp. 62–67.
- [10] E. Gençay, P. Schüller, E. Erdem, Applications of non-monotonic reasoning to automotive product configuration using answer set programming, *Journal of Intelligent Manufacturing* 30 (2019) 1407–1422. doi:10.1007/s10845-017-1333-3.
- [11] J. Tiihonen, M. Heiskala, A. Anderson, T. Soininen, WeCoTin – A practical logic-based sales configurator, *AI Communications* 26 (2013) 99–131. doi:10.3233/aic-2012-0547.
- [12] J. Tiihonen, A. Anderson, VariSales, in: [3], 2014, pp. 309–318. doi:10.1016/B978-0-12-415817-7.00026-8.
- [13] T. Soininen, J. Tiihonen, T. Männistö, R. Sulonen, Towards a general ontology of configuration, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 12 (1998) 357–372. doi:10.1017/s0890060498124083.
- [14] A. Falkner, A. Ryabokon, G. Schenner, K. Shchekotykhin, OOASP: connecting object-oriented and logic programming, in: F. Calimeri, G. Ianni, M. Truszczyński (Eds.), *Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’15)*, volume 9345 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2015, pp. 332–345. doi:10.1007/978-3-319-23264-5_28.
- [15] L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley, K. Wolter, Configuration knowledge representation and reasoning, in: [3], 2014, pp. 41–72. doi:10.1016/b978-0-12-415817-7.00006-2.
- [16] M. Gebser, R. Kaminski, B. Kaufmann, M. Lindauer, M. Ostrowski, J. Romero, T. Schaub, S. Thiele, *Potassco User Guide*, 2 ed., University of Potsdam, 2015. URL: <http://potassco.org>.
- [17] S. Hahn, O. Sabuncu, T. Schaub, T. Stolzmann, clingraph: ASP-based visualization, in: G. Gottlob, D. Inclezan, M. Maratea (Eds.), *Proceedings of the Sixteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’22)*, volume 13416 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2022, pp. 401–414. doi:10.1007/978-3-031-15707-3_31.

Interactive Configuration with ASP Multi-Shot Solving

Richard Comploi-Taupe¹, Andreas Falkner¹, Susana Hahn^{2,3}, Torsten Schaub^{2,3} and Gottfried Schenner¹

¹Siemens AG Österreich, Vienna, Austria

²University of Potsdam, Germany

³Potassco Solutions, Germany

Abstract

The area of product configuration has witnessed a growing demand for systems that can effectively guide users through the configuration process. These systems facilitate interactivity during configuration by combining user actions with automatic solving. In this paper, we present an API that fulfills the basic requirements of interactive configuration. Our implementation is based on the OOASP framework for object-oriented configuration in Answer Set Programming (ASP), leveraging multiple features of the ASP system *clingo* to dynamically introduce components.

1. Introduction

Product configuration has been one of the first successful applications of Answer Set Programming (ASP [1, 2]) [3]. Nonetheless, more than 20 years later, its use in product configurators is still challenging. One open challenge is to allow for interactivity during configuration.

Industrial product configuration deals with large problems. For example, even small infrastructure projects may contain thousands of components and hundreds of component types. Such configurations are typically solved step-by-step by combining interactive actions with automatic solving of sub-problems [4]. Configurator users, such as engineers and sales people, expect a system that guides them through the configuration process. Domain experts provide the configuration model that defines such a process and system.

Using a grounding-based formalism like ASP in this context introduces the risk of a grounding bottleneck [5] due to the large number of required components for satisfying all requirements. The required domain size can vary significantly and is not known beforehand, which leads to the necessity of dynamically introducing new components during the configuration process.

In this work, we present an Application Programming Interface (API) to satisfy basic requirements for interactive configuration [4]. Our implementation is based on OOASP [6], a framework for representing object-oriented

configurations in ASP. Additionally, we exploit multiple features of the ASP system *clingo*¹ [7] to provide interactive functionalities.

After covering background on ASP, product configuration and OOASP, and on our running example in Section 2, we introduce our approach in detail in Section 3. The paper concludes with a discussion in Section 4.

2. Background

2.1. Answer set programming

A logic program consists of rules of the form

$$a_1; \dots; a_m \leftarrow a_{m+1}, \dots, a_n, \\ \quad \quad \quad \text{not } a_{n+1}, \dots, \text{not } a_o.$$

where each a_i is an atom of form $p(t_1, \dots, t_k)$ and all t_i are terms, composed of function symbols and variables. For $1 \leq m \leq n \leq o$, atoms a_1 to a_m are often called head atoms, while a_{m+1} to a_n and $\text{not } a_{n+1}$ to $\text{not } a_o$ are also referred to as positive and negative body literals, respectively. An expression is said to be ground, if it contains no variables. As usual, not denotes (default) negation. A rule is called a fact if $m = n = o = 1$, normal if $m = 1$, and an integrity constraint if $m = 0$. In what follows, we deal with normal logic programs only, for which m is either 0 or 1. Semantically, a logic program induces a set of stable models, being distinguished models of the program determined by the stable models semantics [8].

To ease the use of ASP in practice, several extensions have been developed. First of all, rules with variables are viewed as shorthands for the set of their ground instances. Further language constructs include conditional literals and cardinality constraints [9]. The former are

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

✉ richard.taupe@siemens.com (R. Comploi-Taupe);

andreas.a.falkner@siemens.com (A. Falkner);

hahnmartinlu@uni-potsdam.de (S. Hahn);

torsten@cs.uni-potsdam.de (T. Schaub);

gottfried.schenner@siemens.com (G. Schenner)

👤 0000-0001-7639-1616 (R. Comploi-Taupe); 0000-0002-2894-3284

(A. Falkner); 0000-0003-2622-2632 (S. Hahn); 0000-0002-7456-041X

(T. Schaub); 0000-0003-0096-6780 (G. Schenner)

(CC BY 4.0) © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://potassco.org/clingo>

of the form² $a : b_1, \dots, b_m$, the latter can be written as³ $s \{d_1; \dots; d_n\} t$, where a and b_i are possibly negated (regular) literals and each d_j is a conditional literal; s and t provide optional lower and upper bounds on the number of satisfied literals in the cardinality constraint. We refer to b_1, \dots, b_m as a condition. The practical value of both constructs becomes apparent when used with variables. For instance, a conditional literal like $a(X) : b(X)$ in a rule's body expands to the conjunction of all instances of $a(X)$ for which the corresponding instance of $b(X)$ holds. Similarly, $2 \{a(X) : b(X)\} 4$ is true whenever at least two and at most four instances of $a(X)$ (subject to $b(X)$) are true. More sophisticated examples are given in Section 3.

A particular convenience feature are anonymous variables, denoted uniformly by an underscore ‘_’. Each underscore in a rule is interpreted as a fresh variable. In turn, atoms with anonymous variables are replaced by new atoms dropping these variables; the new atoms are then linked to the original ones by rules expressing projections.

Multi-shot solving allows for solving continuously changing logic programs in an operative way. In *clingo*, this can be controlled via an API for implementing reactive procedures that loop on grounding and solving while reacting, for instance, to outside changes or previous solving results. This is supported by two directives. First, a program can be partitioned into several subprograms by means of the directive `#program`; it comes with a name and an optional list of parameters. Such subprograms can then be grounded upon demand and added to the solver. Second, `#external` directives allow for declaring atoms whose truth value can be set via the API and/or rules that may be added later on. This allows us to continuously assemble ground rules evolving at different stages of a reasoning process and to change program behavior by manipulating the truth values of external atoms via the API.

Full details on the input language of *clingo* along with various examples can be found in the *Potassco User Guide* [10].

2.2. Product Configuration and OOASP

Product Configuration as an activity produces the specification of an artifact that is assembled from instances of given component types and that conforms to a given set of constraints between those components. Component types can have attributes, thus components can be parametrized. Furthermore, components are related to each other via part-of or is-a relationships [11]. In most

configuration problems, a dynamic number of components plays an important role [12].

OOASP⁴ [6, 13] is an ASP-based framework to encode and reason about object-oriented problems such as configuration problems. It defines a Domain Description Language (DDL) specific to the domain of object-oriented models that can be represented by a modelling language corresponding to a UML class diagram. OOASP-DDL defines ASP predicates to encode models (classes, subclass relations, associations, and attributes) and instantiations (instances, is-a relations, instance-level associations, and attribute values). Furthermore, it provides a uniform way to encode (built-in and user-specific) constraints.

Table 1 shows the OOASP-DDL predicates for the encoding of models, and Table 2 shows the OOASP-DDL predicates for the encoding of instantiations.⁵

OOASP constraints are defined using the predicate `ooasp_cv` (“cv” stands for “constraint violation”). Rules with head atoms of this predicate are used instead of ASP constraints to enable configurations to be *checked*, i.e., to derive which constraints are violated in a given configuration (Listing 4). To enforce a configuration to be consistent, a simple ASP constraint forbidding `ooasp_cv` to be true can be added. An `ooasp_cv` atom contains four terms: a unique constraint identifier, the identifier of the faulty object, a string containing a message describing the issue, and a list of additional explanatory terms.

OOASP distinguishes integrity constraints from domain-specific constraints. The former are defined in the OOASP framework itself and refer to issues such as invalid values and violations of association cardinalities. Domain-specific constraints can be defined by a user of OOASP in the same format.

An instantiation (configuration) defined by the predicates from Table 2 is complete if every object is an instance of an instantiable class, and it is correct if no constraint violations can be derived from it. We follow the convention that only leaf classes (i.e., classes that have no subclasses) are instantiable, so every object must be an instance of a leaf class in a complete configuration.

Configuration is usually an interactive task, iteratively involving user interactions (decisions) and automatic reasoning by a solver, e.g., an ASP solver [4]. The goal of our work is to support interactive configuration in a framework based on OOASP, because we think that its natural way of representing subclasses, parts hierarchies, rich element properties, and dynamically created configuration instances allows for understandable and precise product modeling.

²In rule bodies, they are terminated by ‘;’ or ‘.’ [10].

³More elaborate forms of aggregates are obtained by explicitly using function (e.g. `#count`) and relation symbols (e.g. `<=`) [10].

⁴<https://github.com/siemens/OOASP>

⁵We here present a version of OOASP-DDL that has already evolved from the original definition [6] and that has also been slightly simplified for this paper.

Table 1
OOASP-DDL predicates for the encoding of models

ooasp_class(C)	C is a class
ooasp_subclass(SubC, SupC)	SubC is a subclass of SupC
ooasp_assoc(A, C1, C1Min, C1Max, C2, C2Min, C2Max)	A is an association in which each instance of the class C1 is associated to between C2Min and C2Max instances of class C2, and each instance of C2 is associated to between C1Min and C1Max instances of C1.
ooasp_attr(C, A, T)	A is an attribute of class C with type T
ooasp_attr_enum(C, A, D)	D is an element of the domain of attribute A of class C

Table 2
OOASP-DDL predicates for the encoding of instantiations

ooasp_isa(C, O)	O is an object of class C
ooasp_isa_leaf(C, O)	O is an object of leaf class C
ooasp_associated(A, O1, O2)	Object O1 is associated to object O2 in association A
ooasp_attr_value(A, O, V)	The attribute A of object O has value V

2.3. Running example

We use a typical hardware racks configuration problem as the running example for this paper. For easier comparison with non-incremental OOASP the running example is an extension of the racks configuration paper used in the original OOASP paper [6]. The UML class diagram (Figure 1) shows all concepts and relations of the racks knowledge base. This diagram was automatically generated by our Interactive API in integration with *clingo* [14] using a visualization encoding.

Additionally to the constraints implied by the UML diagram, the following constraints hold for the domain:

- An ElementA/B/C/D requires exactly 1/2/3/4 objects of type ModuleI/II/III/IV
- Instances of ModuleI/II/III/IV must be required by exactly one Element
- A SingleRack/DoubleRack has exactly 4/8 Frames
- A Frame containing a ModuleII must also contain exactly one ModuleV

The running example captures the essence of a typical configuration knowledge base in an industrial setting. Of course, real life industrial knowledge bases are much larger (>100 classes, associations, attributes). And the constraints of the domain will vary considerably depending on additional requirements imposed by the customer, regulations, geographic location, etc. Notice that the knowledge base does not contain any restrictions on the number of objects in a configuration or on the order in which objects must be created.

Another property of these knowledge bases is that the number of objects required for a solution is not known beforehand. For example, suppose the user interactively created 5 objects of type ModuleI. The user could assign those modules to the same frame and assign the frame to

a rack. Or the user could assign the modules to different frames and assign those to different racks. In any case, a rack must be connected to at least four frames. Therefore, the first configuration has 10 objects (5 modules, 4 frames, 1 rack), while the second, equally valid configuration has 30 objects (5 modules, 20 frames, 5 racks).

3. Interactive Configurator

Our Configuration API (CAPI) is implemented using Python, relying heavily on multiple features provided by *clingo*'s Python API, as well as the systems *clorm*⁶ and *clingo*. *Clorm* is a Python library providing an Object Relational Mapping (ORM) interface to *clingo*, which we use to map the OOASP predicates defining the knowledge base and the configuration into Python classes. These elements are then visualized as graphs (resembling UML diagrams) using *clingo*. For interactive configuration, we created a scientific prototype User Interface (UI) using *ipywidgets* that employed our CAPI functionalities.

The basic idea behind our approach is to modularize the encodings so that the program can be built incrementally as the number of instantiated objects in the configuration increases based on user interaction. To that end, we use the multi-shot capabilities of *clingo* to solve these continuously changing logic programs. This approach avoids re-grounding and benefits from learned constraints by grounding and solving on demand. More specifically, we defined subprograms that depend on the identifier of each newly introduced object, namely `new_object`. Therefore, whenever the domain size is extended by a new object, all the rules referring to this object are grounded. In this sense, our implementation

⁶<https://github.com/potassco/clorm>

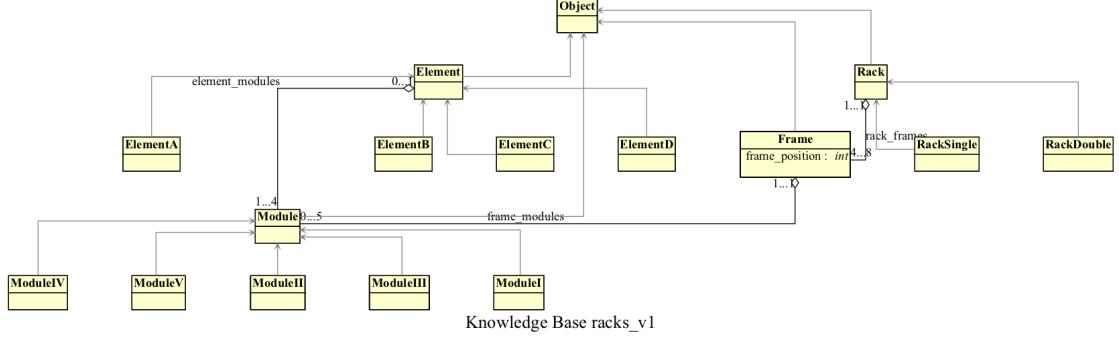


Figure 1: Class diagram for the racks knowledge base generated by *clingraph*.

differs from the previous work [15], in which subprograms were subject to domain-specific actions.

3.1. Interactive tasks

We introduce eight fundamental interactive tasks, derived from [4] and adapted to our multi-shot setting, allowing users to edit a partial configuration \mathcal{C}_P and construct a complete configuration \mathcal{C}_C . First, the user can modify \mathcal{C}_P through the following interactive tasks:

- T1. Setting and un-setting the type of an existing object.
- T2. Adding and removing associations between two objects.
- T3. Setting and un-setting values for attributes.

Such tasks are done using external atoms in *clingo*, so that no re-grounding is required. Due to lack of space, we will focus only on the encoding of task T3. Tasks T1 and T2 are encoded in a similar way. We show in Listing 1 how the user input is handled for task T3. Line 1 corresponds to *clingo*'s program directive indicating that the subprogram depends on `new_object`. Thus, all the following rules will be grounded on demand when a new object is introduced. Lines 2 and 3 define an external atom `user(ooasp_attr_value(A,new_object,V))` for each attribute A and value V of the `new_object`. Notice that we need to generate all possible combinations since the object can be assigned to any class. The truth value of these externals will be set based on the user's selection. Finally, the rule in Lines 4 and 5 makes sure that if the user selected a value for an attribute it will be considered in the encoding.

- T4. Extending the configuration with a new object.

As mentioned before, the grounding of subprograms will exclusively occur when the user performs task T4. Consequently, in the rest of the tasks the number of objects will remain fixed. Note that the newly introduced

```

1  #program domain(new_object).
2  #external user(ooasp_attr_value(A,new_object,V)):
3      ooasp_attr_enum(_,A,V).
4  ooasp_attr_value(A,new_object,V) :- 
5      user(ooasp_attr_value(A,new_object,V)).

```

Listing 1: User input

object will not have a type; its type will be set explicitly by the user with T1 or by the system with T5.

Finally, we identified three reasoning tasks in which solving is necessary.

- T5. Using the current objects to generate \mathcal{C}_C from \mathcal{C}_P via choice rules.⁷
- T6. Checking if \mathcal{C}_P is complete or if it violates any constraints.
- T7. Obtaining the list of available edit-options for the user via brave reasoning.

These tasks are distinguished within the encoding via externals. The truth values of these externals is controlled internally depending on the task selected by the user. In this case, one external atom `guess` states that the guessing of objects' types, associations and values is active. Two additional externals `check_permanent_cv` and `check_potential_cv` activate the integrity constraints for the two types of constraints. Constraints are divided in this way, since we need to take into consideration that we are building \mathcal{C}_C in an interactive and incremental way, therefore some of these constraints might be violated on \mathcal{C}_P but fixed once \mathcal{C}_C is reached. The intuition for this decision can be taken from the fields of Runtime Verification and Monitoring, where a constraint is either *satisfied*, *potentially violated* (might or might not remain a violation in the future) or *permanently violated* (a violation in all possible futures). *Potential constraint violations* are those that can potentially be fixed by adding more information in a later stage of the process.

⁷A choice rule is a rule with a cardinality constraint in the head.

```

1  1 { ooasp_attr_value(A,new_object,V):
2      ooasp_attr_enum(C,A,V) } 1 :- 
3      ooasp_isa(C,new_object),
4      ooasp_attr(C,A,T),
5      ooasp_attr_enum(C,A,_),
6      guess.

```

Listing 2: Choice rule to guess the value of an attribute

```

1  :- ooasp_cv(CV,_,_,_),
2      not ooasp_potential_cv(CV),
3      check_permanent_cv.
4  :- ooasp_cv(CV,_,_,_),
5      ooasp_potential_cv(CV),
6      check_potential_cv.

```

Listing 3: Integrity constraints enforcing constraint violations

For instance, a lower bound of an association that has not been reached, or a value that is missing. These constraints are identified in the encoding with the predicate `ooasp_potential_cv`. On the other hand, *permanent constraint violations* refer to violations that can no longer be fixed, such as upper bounds of an association or an attribute value of a wrong type.

For task T5, `guess`, `check_permanent_cv` and `check_potential_cv` are set to true in order to find a complete and valid configuration \mathcal{C}_C . This is achieved by using choice rules to generate possible values, types and associations for the objects, which are activated by the external `guess`. If no \mathcal{C}_C can be found with the current number of objects the result of the task will be UNSATISFIABLE. To illustrate this, Listing 2 shows the choice rule to select a value for an attribute. The rule can be read as follows: if the new object is of type C (Line 3), where type C has an attribute A (Line 4) with some elements in the domain (Line 5), and the guessing is active (Line 6), then out of all the possible values for A choose a single value V. Notice that this rule is also grounded incrementally and uses the corresponding `new_object`.

For task T6, we set the externals `check_permanent_cv` and `check_potential_cv` to false so that all the `ooasp_cv` atoms are part of the computed stable model, thus deriving the issues with \mathcal{C}_P . In Listing 3 we show the integrity constraints handling the constraint violations, which are also grounded incrementally. Lines 1 to 3 make sure that no constraint violation is derived if the external `check_permanent_cv` is true and the constraint violation CV is not a potential but a permanent one. Similarly, the second constraint (Lines 4 to 6) enforces potential constraints when `check_potential_cv` is true.

For task T7, we want to provide the user with valid actions from T1, T2 and T3. To achieve this, potential constraints are ignored by setting the external `check_potential_cv` to false so that we allow con-

```

1  ooasp_potential_cv(no_val).
2  ooasp_cv(no_val,new_object,"Missing value for {}",(A)) :- 
3      ooasp_attr(C,A,T),
4      ooasp_attr_enum(C,A,_),
5      ooasp_isa(C,new_object),
6      not ooasp_attr_value(A,new_object,_).

```

Listing 4: Constraint violation of a missing value

straints of this type to be violated in \mathcal{C}_P while still getting a satisfiable answer. However, the permanent constraints should remain active since we want to discard anything that can't be fixed by further interaction with the system. With this set, we use the brave reasoning capabilities of `clingo` to obtain the union of all stable models, and thus, all the possible options for types, values of attributes, and associations.

As before, we use the attribute values to exemplify the use of task T7 in Listing 4. The rule in Lines 2 to 6 derives the constraint violation `no_val` of having no value set for an attribute. As expected, this is a potential constraint (expressed in Line 1) since the user can later on select the missing value. The constraint violation is then derived for any attribute of the `new_object` that has no corresponding value assigned via `ooasp_attr_value`.

Some other checks might depend on values that have to be recomputed on every grounding step, such as the arity of an association. In other words, if an aggregate `#count` is used to compute the objects associated to `new_object`, it will only count the objects that are already grounded at that time. This means that the arity computed in previous steps must be disregarded. Therefore, we need an additional external `active(new_object)` that indicates the current step to know if the aggregate's value is older and thus expired. Notice that the current step corresponds to the object identifier that is being grounded at that time.

T8. Extend \mathcal{C}_P incrementally to generate \mathcal{C}_C

Given all these functionalities, finding the smallest \mathcal{C}_C that extends \mathcal{C}_P can be encapsulated into the combined task T8. The program for task T8 will proceed following an incremental approach: \mathcal{C}_P is extended with a new object (T4) and then tries to generate \mathcal{C}_C (T5), these steps are repeated until a \mathcal{C}_C is found.

3.2. Performance

Knowing about the huge solution space, we improved efficiency right from the beginning by including symmetry breaking constraints which get rid of multiple symmetric configurations. The two symmetries identified can be found in Listing 5. Both symmetries correspond to permutations of the classes assigned to objects. The constraint in Lines 1 to 6 ensures that the classes assigned to objects smaller than the `new_object`

```

1 :- ooasp_isa_leaf(C1,new_object),
2     ooasp_isa_leaf(C2, ID),
3     ID<new_object,
4     C1<C2,
5     not user(ooasp_isa_leaf(C1,new_object)),
6     not user(ooasp_isa_leaf(C2, ID)).

8 :- ooasp_isa_leaf(_,new_object),
9     not ooasp_isa_leaf(_, ID),
10    ooasp_isa(_, ID),
11    ID<new_object.

```

Listing 5: Symmetry breaking constraints

are also smaller. Notice that this is only applied to decisions made by the solver, excluding assignments made by the user (Lines 5 and 6). Otherwise the user setting the class of an object (via T1) could lead to unsatisfiability. The constraint in Lines 8 to 11 makes sure that any objects left out from the configuration (with no class assigned) are always those with larger ids. For instance, the assignment $\{(1, C_1), (2, \text{undef}), (3, C_2)\}$ would be removed by the second constraint in favor of $\{(1, C_1), (2, C_2), (3, \text{undef})\}$. Similarly, $\{(1, C_1), (2, C_2), (3, C_1)\}$ would not be valid due to the first constraint, keeping the symmetric assignment $\{(1, C_1), (2, C_1), (3, C_2)\}$.

We performed some empirical tests in the system to check the performance based on the running example from Section 2.3. The aim of the first test was to generate a \mathcal{C}_C of size 41 with one RackSingle associated to four Frames, each Frame with four associated Modules with one corresponding Element. First, we extended \mathcal{C}_P with 41 objects via T4 where 18 of those objects were selected to be of the Element class. Then we used task T5 to find our expected \mathcal{C}_C . Overall, these steps took 7 seconds of grounding time and 3 seconds of solving. At this point, any of the tasks T1, T2, T3, and T6 can be done without delay. However, obtaining the list of options with task T7 didn't finish within 5 minutes. This happened since the number of valid options is quite large when a user has 23 objects without an associated class. In practice we expect this to decrease with a more tightly defined \mathcal{C}_P during the interaction. As a second test, we analyzed task T8 by creating n Element instances and finding incrementally the corresponding \mathcal{C}_C . The results can be found in Figure 2a for $n \in \{8, 9, 10\}$. When we increased n to 11, the task didn't finish within a 5 minute time out. Looking closely at the performance for $n = 9$ in Figure 2b, we can see that the issue lies on having to prove unsatisfiability for domain sizes 9 to 22. Proving unsatisfiability implies going through all the search space to make sure there is no answer, which is quite costly as the domain increases. In our test, this is the case when trying to find a non-existing \mathcal{C}_C with a domain size of 22 objects. This was not the case in our previous test where we have all the required objects to obtain a satisfiable

answer in a single call, comparable to the solving time taken for domain size 23 in Figure 2b.

3.3. User Interface Prototype

To give an impression of our prototype, we include screenshots (Figures 3,4) of the UI rendered in a Jupyter notebook for the racks examples from Section 2.3. This UI has 6 sections. The section on the upper left corner shows the current \mathcal{C}_P using *clingo*. To the right, the history of actions taken by the user is rendered as a list in the **History** section. In our example, the user started with the first two actions being the extension of the domain by two new objects. This is done via the button in the **Extend** section below, corresponding to task T4. Then, in steps 3 and 4, the user assigned classes to objects 1 and 2, respectively, using the **Edit** section. This section employs T7 to generate a dropdown for each object with the list of possible options (T1, T2, T3). Step 5 corresponds to T6, triggered by clicking the button in the **Check** section. This action prints in red the constraint violations found for each object. In this case, Object 2 (the frame) violates the missing value constraint from Listing 4 and the lower-bound constraint, since it should be associated with one rack. Similarly, Object 1 (the instance of RackSingle) is violating the lower-bound constraint, as well as a domain-specific constraint enforcing it to be associated to exactly 4 frames. For the last task, namely T5, the user must work on the **Browse** section of the UI. By clicking the button labeled "Next solution", the system would perform task T5. However, since there is no \mathcal{C}_C with the current number of objects it would get an error. As mentioned before, this is overcome by extending \mathcal{C}_P incrementally, which is done by task T8 in step 6 when the user clicks on "Find incrementally" (Figure 4). As a consequence, the system will internally find \mathcal{C}_C and render it in the bottom right. In this *clingo* image, the values from \mathcal{C}_P will appear in green. As a next step, the user could either browse through all the possible \mathcal{C}_C of the same size, or select the current \mathcal{C}_C as the new \mathcal{C}_P to be further edited. Notice that while browsing, no options are shown in the **Edit** section. This happens since we have a single *clingo* control object which is currently in the middle of solving, thus, it can't generate the brave consequences of T7.

4. Discussion

In this paper, we have presented an interactive configurator that enables engineers and salespeople, among other configurator users, to incrementally build configurations. Our contribution involves the development of an API built upon the OOASP framework and the *clingo* system. The OOASP framework provided us with a domain

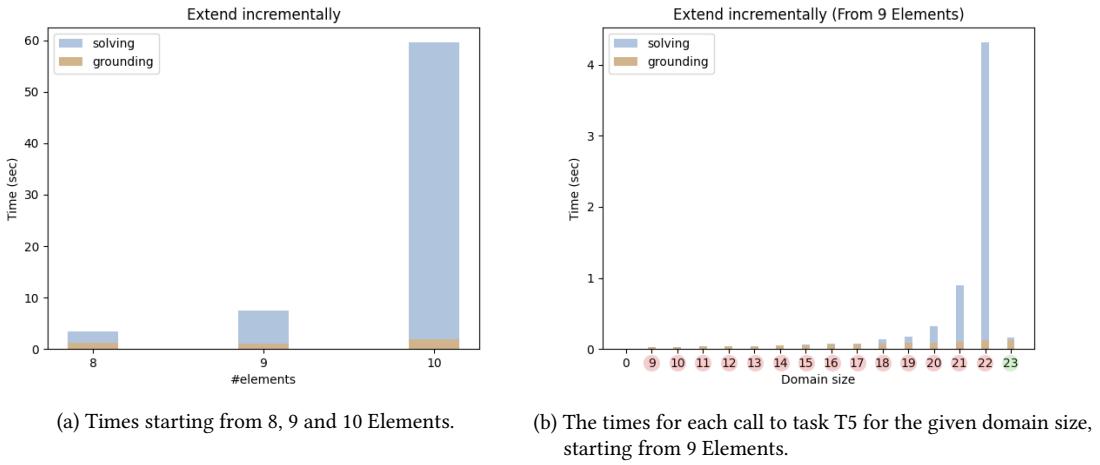


Figure 2: Times for task T8 (Extending incrementally)

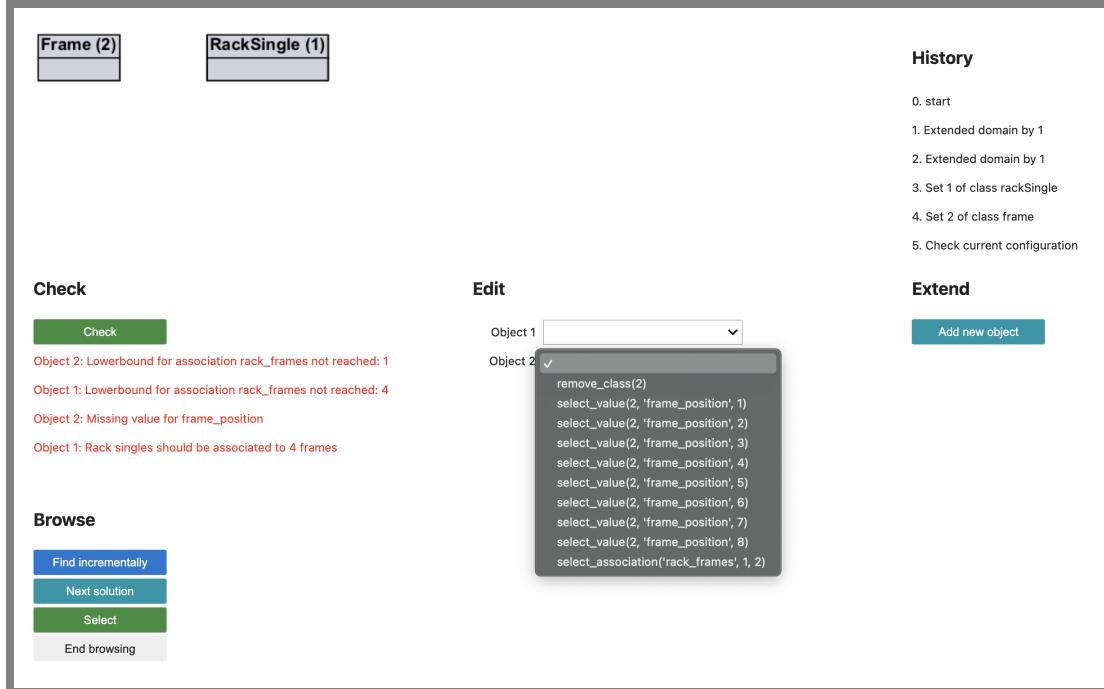


Figure 3: UI for interactive configuration generated with *ipywidgets* rendered in a Jupyter notebook.

description language to encode models, instances, and solutions, while *clingo*'s multi-shot capabilities allowed us to dynamically extend the configuration by adding components on demand. The integration of this multi-shot approach to interactive configuration distinguishes our approach from previous work.

To fulfill the basic requirements of interactive configuration, we identified and implemented eight distinct tasks, which we described in detail. To demonstrate the functionality of our API, we developed a prototype user interface and showcased the step-by-step creation of a configuration using our running example. As the UI is

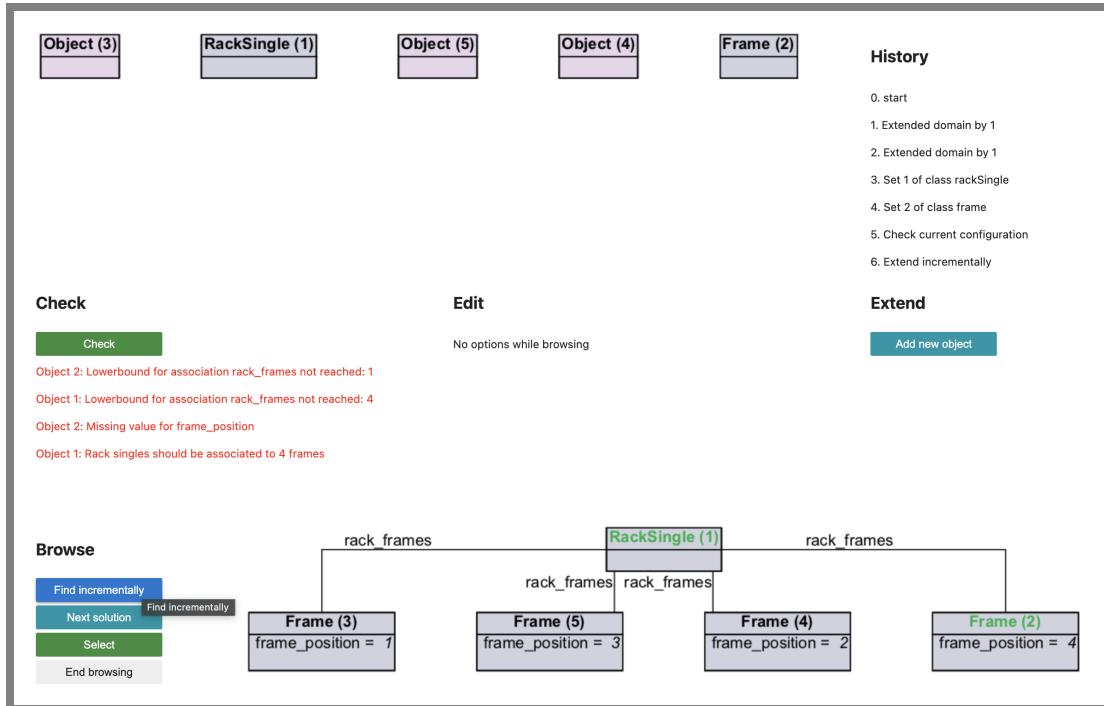


Figure 4: UI for interactive configuration generated with *ipywidgets* rendered in a Jupyter notebook. Where the user is browsing the extensions for \mathcal{C}_P into \mathcal{C}_C .

currently in a prototypical stage, gathering real user feedback remains a future goal for subsequent versions of the system. To assess the performance of the system, we conducted empirical tests and identified the need for further improvements. In particular, we boosted efficiency by incorporating symmetry-breaking constraints.

However, we also encountered performance issues with our incremental approach when dealing with larger instances, which warrants future research. More specifically, we plan to explore alternative methods for extending the configuration beyond the one-by-one incremental process. For instance, we are interested in investigating scheduling techniques [16] and pre-computing the minimal number of required objects [17]. Additionally, we aim to enhance usability by incorporating additional advanced features, such as linear domains, which enable more sophisticated reasoning in the configuration process.

References

- [1] V. Lifschitz, Answer Set Programming, 2019. doi:10.1007/978-3-030-24658-7.
- [2] M. Gelfond, Y. Kahl, Knowledge Representation, Reasoning, and the Design of Intelligent Agents:

The Answer-Set Programming Approach, Cambridge University Press, New York, NY, USA, 2014.

- [3] T. Soininen, I. Niemelä, J. Tiuhonen, R. Sulonen, Representing configuration knowledge with weight constraint rules., in: A. Provetti, T. Son (Eds.), Proceedings of the AAAI Spring Symposium on Answer Set Programming (ASP'01), AAAI/MIT Press, 2001, pp. 195–201. URL: <http://www.cs.nmsu.edu/%7Etson/ASP2001/20.ps>.
- [4] A. Falkner, A. Haselböck, G. Krames, G. Schenner, H. Schreiner, R. Taupe, Solver requirements for interactive configuration. 26 (2020) 343–373. doi:10.3897/jucs.2020.019.
- [5] T. Eiter, W. Faber, M. Fink, S. Woltran, Complexity results for answer set programming with bounded predicate arities and implications, Ann. Math. Artif. Intell. 51 (2007) 123–165. doi:10.1007/s10472-008-9086-5.
- [6] A. Falkner, A. Ryabokon, G. Schenner, K. Shchekotykhin, OOASP: connecting object-oriented and logic programming, in: F. Calimeri, G. Ianni, M. Truszczyński (Eds.), Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15), volume 9345 of *Lecture Notes in Artificial Intelligence*

- gence, Springer-Verlag, 2015, pp. 332–345. doi:10.1007/978-3-319-23264-5_28.
- [7] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, *Theory and Practice of Logic Programming* 19 (2019) 27–82. doi:10.1017/S1471068418000054.
- [8] M. Gelfond, V. Lifschitz, Logic programs with classical negation, in: D. Warren, P. Szeredi (Eds.), *Proceedings of the Seventh International Conference on Logic Programming (ICLP'90)*, MIT Press, 1990, pp. 579–597.
- [9] P. Simons, I. Niemelä, T. Soininen, Extending and implementing the stable model semantics, *Artificial Intelligence* 138 (2002) 181–234.
- [10] M. Gebser, R. Kaminski, B. Kaufmann, M. Lindauer, M. Ostrowski, J. Romero, T. Schaub, S. Thiele, Potassco User Guide, 2 ed., University of Potsdam, 2015. URL: <http://potassco.org>.
- [11] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen (Eds.), *Knowledge-Based Configuration – From Research to Business Cases*, Morgan Kaufmann, Boston, 2014. doi:10.1016/B978-0-12-415817-7.00029-3.
- [12] A. A. Falkner, G. Friedrich, A. Haselböck, G. Schenner, H. Schreiner, Twenty-five years of successful application of constraint technologies at siemens, *AI Mag.* 37 (2016) 67–80. doi:10.1609/aimag.v37i4.2688.
- [13] A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. Teppan, Industrial applications of answer set programming, *Künstliche Intelligenz* 32 (2018) 165–176. doi:10.1007/s13218-018-0548-6.
- [14] S. Hahn, O. Sabuncu, T. Schaub, T. Stolzmann, clingraph: ASP-based visualization, in: G. Gottlob, D. Inglezani, M. Maratea (Eds.), *Proceedings of the Sixteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LP-NMR'22)*, volume 13416 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2022, pp. 401–414. doi:10.1007/978-3-031-15707-3_31.
- [15] R. Comploi-Taupe, G. Francescutto, G. Schenner, Applying incremental answer set solving to product configuration, in: *Proceedings of the 26th ACM International Systems and Software Product Line Conference – Volume B*, Association for Computing Machinery, New York, NY, USA, 2022, pp. 150–155. doi:10.1145/3503229.3547069.
- [16] Y. Dimopoulos, M. Gebser, P. Lühne, J. Romero, T. Schaub, plasp 3: Towards effective ASP planning, in: M. Balduccini, T. Janhunen (Eds.), *Proceedings of the Fourteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'17)*, volume 10377 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2017, pp. 286–300. doi:10.1007/978-3-319-61660-5_26.
- [17] M. Aschinger, C. Drescher, G. Gottlob, H. Vollmer, Loco – A logic for configuration problems, *ACM Trans. Comput. Log.* 15 (2014) 20:1–20:25. doi:10.1145/2629454.

PERFECT: PErformant and Robust read-to-fly FIetT ConfiguraTion: from Robot to Mission Plan*

Elise Vareilles^{1,*†}, Stéphanie Roussel^{2,†} and Gauthier Picard^{2,†}

¹ISAE SUPAERO, University of Toulouse, 10, avenue Édouard-Belin BP 54032 - 31055 Toulouse CEDEX 4, France

²ONERA / DTIS, University of Toulouse, 2 Av. Edouard Belin, 31000 Toulouse, France

Abstract

With the increasing autonomy of aerial, ground and underwater robots, fleets of robots are now being used for many types of missions, such as exploration, rescue, disaster relief or civil and military security. Some of these applications require fleets of heterogeneous robots, i.e., with different capabilities, different means of mobility and different equipment, which may or may not be coordinated autonomously to carry out the missions for which the fleet is dedicated. The problem of multi-level configuration of a fleet of heterogeneous robots and the scientific issues raised by such a problem are explored in this short article.

Keywords

Configuration, Multi-agent Systems, Operational Research, Performance, Robustness

1. Introduction

We present here a prospective application of configuration to a heterogeneous robot fleet (or swarm). This very problem has been the subject of a joint project between ONERA Toulouse France and ISAE SUPAERO France.

The structure of the article is as follows. The context of our study topic is presented in Section 2. Then, because multi-level configuration is a complex and quiet new field, some open research questions are presented in Section 3.

2. Background and Research Statement

With the increasing autonomy of aerial, ground and underwater robots, fleets of robots are now being used for many types of missions. Examples include package delivery, flying taxis, field exploration, rescue and disaster relief. More and more applications require fleets of heterogeneous robots, e.g., with different capabilities such as *detect*, *communicate*, *observe*, *move*, etc. For example, an exploration mission may require the collaboration of

ground robots with at least the ability to *move* and *communicate*, and aerial robots with at least the ability to *observe* and *communicate*. The success of a multi-robot mission depends, among other things, on the configuration of the fleet carrying out the mission [1].

This article examines the problem of multi-level configuration of robot fleets. By multi-level configuration, we mean the simultaneous configuration of each robot (first layer) and the robot fleet itself (second layer) in order to perform the dedicated missions in a high-performance and robust manner (third layer). That multi-level configuration problem requires an analysis of the relationships between these three levels of configuration, both upstream in fleet composition and downstream in fleet operation.

By *configuration*, we mean:

1. for each *robot*: the selection of its equipment and capabilities,
2. for the *robot fleet*: its composition, i.e., the number and type of each robot, and its layout. By layout, we mean the architecture of the swarm: cloud, diamond, rung refused, etc.
3. and for the *missions*: the set of missions that the robot fleet can perform by its reconfiguration.

This multi-level configuration problem raises numerous research issues, such as (1) the representation/modelling of the configuration knowledge (compact modelling language), (2) the elicitation of constraints (what is allowed or forbidden) and criteria (what is preferred) that apply both to the fleet configuration and to each robot in it, and (3) the development of algorithms to generate optimal or at least good quality solutions.

It is generally expected that a robot fleet performs well and is robust during the mission execution. For example,

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

†These authors contributed equally.

✉ elise.vareilles@isae-supaero.fr (E. Vareilles); stephanie.roussel@onera.fr (S. Roussel); gauthier.picard@onera.fr (G. Picard)
🌐 https://pagespro.isae-supaero.fr/elise-vareilles/ (E. Vareilles); https://onera.academia.edu/SRoussel (S. Roussel); https://gauthier-picard.info/ (G. Picard)
👤 0000-0001-6269-8609 (E. Vareilles); 0000-0001-7033-555X (S. Roussel); 0000-0002-9888-9906 (G. Picard)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

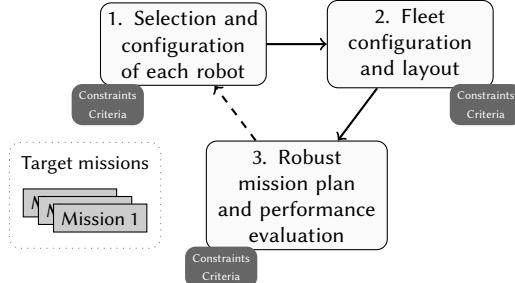


Figure 1: Multi-level configuration problem steps

for a parcel delivery mission, the performance of the fleet can be defined by the time required to complete all deliveries, and its robustness can be defined by the ability of the fleet to complete the mission despite the failure of one or more robots, with the least loss of performance. Assessing the performance and/or robustness of a fleet for a given configuration generally involves generating one or more mission plans for the fleet and then analysing the metrics associated with these plans. The generation of such plans is a combinatorial problem by itself.

Here a mission plan consists in the allocation of the different mission tasks to the robots and their scheduling, meeting the constraints (time, resource availability, etc.) and optimising the performance and/or robustness criteria (mission duration, minimisation of resources used, contingency management, etc.). For example, determining the minimum time required to deliver parcels using a given number of robots whose capacities are fixed *a priori* is a hard problem (NP-complete problem of vehicle rounds [2])). Similarly, planning problems with the presence of uncertainty constitute a vast subject of research ([3, 4]).

In practice, the performance and robustness of a fleet for its mission is often approximated at the time of configuration. It is only after the fleet is configured that a powerful, robust plan for the mission is generated, allowing performance and robustness to be assessed in detail. However, this sequential aspect can lead to sub-optimal solutions. As illustrated in Fig. 1, the performance and robustness analysis can lead to a modification of some robot configuration, which implies going through a whole new configuration cycle. In the general case, there can be numerous iterations before reaching satisfactory results for each configuration level. For example, an undersized or poorly configured fleet can have a significant impact on mission performance or even cause a mission to fail. On the other hand, some missions may require specific fleet configurations and/or specific robot configurations to meet performance objectives. For instance, a rescue mission may require specific communication capabilities for certain robots acting as routers. Note that some re-

search deals simultaneously with the configuration of a fleet and its optimal planning ([5, 6, 7, 8]). However, the associated configuration problem is weakly combinatorial in the sense that it is possible to enumerate all configurations at the time of mission planning.

In this work, we aim to address the performance and robustness constraints and criteria right from the fleet configuration phase, in cases where the configuration is complex. More specifically, given one or more mission types for the fleet with target performance and robustness and associated robot capabilities, a set of possible equipment with their compatibility constraints and the relationships between equipment and robot capabilities, we aim to define the number of robots that make up the fleet and the configuration of each robot so as to achieve the desired performance and robustness targets. The problem then consists in exploring the space of configurations, guided by the performance and robustness evaluation.

This problem can be approached in several ways. First, there is the question of how to express knowledge, constraints and preferences, both from the point of view of fleet configuration and from the point of view of performance and robustness in the context of the [9] mission. Approaches such as constraint programming and multi-agent modelling [10] can be used. Appropriate solution strategies must then be developed. One possible approach would be to take inspiration from bi-level optimisation ([11, 12, 13]) and define one level dedicated to configuration and another to performance or robustness evaluation. The challenge is then to allow the levels to interact and guide each other towards optimal solutions. Several algorithmic approaches can be used: constraint programming, local search, metaheuristics and possible coupling with learning-based strategies. The approaches developed in the project will be validated by experiments on multi-robot problems that are representative of the applications dealt with at ONERA.

3. Research Questions

In order to address the multi-level configuration problem described above, three main scientific questions arise:

- **RQ1: What is the formal modelling of this multi-robot fleet configuration problem?**

More specifically, the knowledge base in input to the problem must contain all the knowledge needed to solve the problem. In particular, all the following elements should be formalized:

- the description of the platforms and the fleet,
- the constraints and objectives associated with this equipment and the fleet,

- the capabilities required to solve the mission and the links between equipment and capabilities,
- definitions of performance and robustness for the mission(s).

It should be noted that it is also possible to consider the organisational aspects of the fleet in the configuration. For instance, a robot fleet for a field exploration mission could be organized following a centralized or decentralized scheme. In the first case, it means that one robot plays a central role and must therefore have the corresponding capacities such as the ability to communicate with all other robots. In the second case, robots must have their own planning decision module and should therefore be equipped accordingly.

- **QS2: What types of approaches and algorithms are effective in solving the multi-robot fleet configuration problem?** This question can be broken down into a number of sub-problems:

1. How can configuration and multi-agent system approaches be combined to define the configuration of the multi-robot fleet?
2. How can robustness and performance indicators be used to guide the configuration of multi-robot fleets?
3. How can the multi-robot fleet and mission plans be defined simultaneously to quickly converge to a good solution?
4. How can we integrate the notion of uncertainty into the different levels of decision making and assess its impact on the quality of the proposed solutions?

These algorithmic strategies may vary depending on whether we are considering a single-criteria or multi-criteria problem, and whether we are searching for optimal or non-optimal solutions. In the case of searching for non-optimal but good quality solutions, there is also the question of the distance to optimality and the calculation of good bounds for evaluating this distance.

- **QS3: How will the proposals be evaluated and validated on realistic case studies?** This will involve not only the implementation of the defined algorithms, but also their evaluation on concrete data sets. Depending on the type of mission under consideration, there will be issues of re-use and adaptation of data sets, or their generation, as well as simulation requirements.

Acknowledgments

The authors would like to thank the ONERA, ISAE-

SUPAERO and ENAC Federation for its support of this work.

References

- [1] S. Mittal, F. Frayman, Towards a generic model of configuraton tasks, in: Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'89, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, p. 1395–1401.
- [2] A. O. Adewumi, O. J. Adeleke, A survey of recent advances in vehicle routing problems, International Journal of System Assurance Engineering and Management 9 (2018) 155–172.
- [3] J. Blythe, An overview of planning under uncertainty, Artificial intelligence today (1999) 85–110.
- [4] T. Chaari, S. Chaabane, N. Aissani, D. Trentesaux, Scheduling under uncertainty: Survey and research directions, in: 2014 International Conference on Advanced Logistics and Transport (ICALT), 2014, pp. 229–234. doi:10.1109/ICAdLT.2014.6866316.
- [5] G. F. List, B. Wood, L. K. Nozick, M. A. Turnquist, D. A. Jones, E. A. Kjeldgaard, C. R. Lawton, Robust optimization for fleet planning under uncertainty, Transportation Research Part E: Logistics and Transportation Review 39 (2003) 209–227. URL: <https://www.sciencedirect.com/science/article/pii/S1366554502000261>. doi:[https://doi.org/10.1016/S1366-5545\(02\)00026-1](https://doi.org/10.1016/S1366-5545(02)00026-1).
- [6] R. F. Lemme, E. F. Arruda, L. Bahiense, Optimization model to assess electric vehicles as an alternative for fleet composition in station-based car sharing systems, Transportation Research Part D: Transport and Environment 67 (2019) 173–196. URL: <https://www.sciencedirect.com/science/article/pii/S1361920918304656>. doi:<https://doi.org/10.1016/j.trd.2018.11.008>.
- [7] R. Pinto, A. Lagorio, R. Golini, Urban freight fleet composition problem, IFAC-PapersOnLine 51 (2018) 582–587. URL: <https://www.sciencedirect.com/science/article/pii/S2405896318315064>. doi:<https://doi.org/10.1016/j.ifacol.2018.08.381>, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.
- [8] Sule Yildirim, B. Yildiz, Electric bus fleet composition and scheduling, Transportation Research Part C: Emerging Technologies 129 (2021) 103197. URL: <https://www.sciencedirect.com/science/article/pii/S0968090X21002126>. doi:<https://doi.org/10.1016/j.trc.2021.103197>.
- [9] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration – From Research to Business Cases, Morgan Kaufmann

- Publishers, United States, 2014. doi:10.1016/B978-0-12-415817-7.00029-3.
- [10] G. Weiss, *Multiagent Systems*, The MIT Press, 2013.
 - [11] C. Lei, W.-H. Lin, L. Miao, A two-stage robust optimization approach for the mobile facility fleet sizing and routing problem under uncertainty, *Computers & Operations Research* 67 (2016) 75–89. URL: <https://www.sciencedirect.com/science/article/pii/S0305054815002178>. doi:<https://doi.org/10.1016/j.cor.2015.09.007>.
 - [12] H. R. Sayarshad, R. Tavakkoli-Moghaddam, Solving a multi periodic stochastic model of the rail–car fleet sizing by two-stage optimization formulation, *Applied Mathematical Modelling* 34 (2010) 1164–1174. URL: <https://www.sciencedirect.com/science/article/pii/S0307904X09002340>. doi:<https://doi.org/10.1016/j.apm.2009.08.004>.
 - [13] P. Pitiot, M. Aldanondo, E. Vareilles, P. Gaborit, M. Djefel, S. Carbonnel, Concurrent product configuration and process planning, towards an approach combining interactivity and optimality, *International Journal of Production Research* 51 (2013) 524–541. URL: <https://doi.org/10.1080/00207543.2011.653449>. doi:10.1080/00207543.2011.653449.
arXiv:<https://doi.org/10.1080/00207543.2011.653449>.

Construction of Decision Diagrams for Product Configuration

Maxim Popov^{1,2}, Tomáš Balyo¹, Markus Iser^{2,3} and Tobias Ostertag^{1,*}

¹CAS Software AG, CAS-Weg 1 - 5, 76131 Karlsruhe, Germany

²Karlsruhe Institute of Technology (KIT), KIT-Department of Informatics, Karlsruhe, Germany

³University of Helsinki, Department of Computer Science / HIIT, Helsinki, Finland

Abstract

Knowledge compilation is a well-researched field focused on translating propositional logic formulas into efficient data structures that allow polynomial-time online queries related to the SAT problem. Knowledge compilation techniques can be used to partition product configuration tasks into two distinct phases: fast online processing and slow offline preprocessing. Binary Decision Diagrams (BDDs) are widely studied in this area and provide a graph representation of Boolean formulas. However, BDD construction can be time-consuming, particularly for large instances, as their size grows exponentially with the number of variables. This paper explores methods to improve BDD construction time, including optimizing variable ordering. The evaluation involves applying these techniques to formulas in Rich Conjunctive Normal Form, comparing the results with Sentential Decision Diagrams. The experiments use CAS Software AG benchmarks.

Keywords

Configuration, Knowledge Compilation, Decision Diagrams

1. Introduction

Propositional logic is a common form of representing real-life logical relations and rules in a way that can easily be used in computer. The following example demonstrates how Boolean formulas are used in the area of product configuration:

Example 1.1. Suppose a company selling bikes offers various configurations, where selecting one component (i.e., bike frame) can limit choices for other components (i.e., wheels) due to compatibility constraints. These constraints can be represented as Boolean formulas.

$$R1 = \neg F1 \vee ((W \vee B) \wedge \neg BL \wedge \neg G) \quad (1)$$

Each variable in Equation 1 is assigned a value of *true* if the option is chosen. $F1$ is the variable representing the frame option, and W, B, G, BL are the variables representing the frame colors white, blue, green, and black respectively. The formula represents the rule that if a frame 1 is chosen, only the colors white and blue can be selected.

Configuration of complex products with many options may be a computationally hard problem that also can be

formulated as a Boolean Satisfiability Problem (SAT). SAT involves determining if a Boolean formula has satisfying assignments. While the problem is NP-complete, it can often be solved online using SAT solvers. Another way is to use knowledge compilation methods, whereby the solutions first get prepared and stored in a data structure offline and then can efficiently be retrieved online. Binary Decision Diagrams (BDDs) and Ordered Binary Decision Diagrams (OBDDs) are well-known knowledge compilation methods that represent Boolean formulas as binary trees. They were used in formal verification and were proved to be efficient in analysing systems with large amount of states [1].

Given the OBDD representation of a Boolean formula, satisfiability can be checked in constant time, solutions can be found in linear time, and models can be counted in polynomial time [2]. However, the size of the BDD as well as its construction time can be exponential in the number of variables.

For the product configuration, it means that configuration rules can be efficiently verified in the runtime, but we have to consider potentially long preprocessing time. Therefore, the reduction of BDD is essential for improving performance and can be achieved by optimizing variable ordering.

In this paper, we overview existing approaches for minimizing BDD size, apply them to RCNF formulas and introduce modifications of existing approaches: variable frequency and M-FORCE constraint ordering heuristics and construction strategies. We evaluate existing as well as our heuristics using real-world configurations and compare them to existing approaches. Lastly, we briefly present our modification of ordering heuristics for Sen-

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

✉ maxim.popov@campus.tu-berlin.de (M. Popov); markus.iser@kit.edu (M. Iser); tobias.ostertag@cas.de (T. Ostertag)
ID 0000-0003-2904-232X (M. Iser); 0000-0003-3294-3807 (T. Ostertag)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). CEUR Workshop Proceedings (CEUR-WS.org)

tential Decision Diagrams (SDDs) construction, which is a recently developed type of decision diagram that is a superset of OBDDs.

Chapter 2 presents some basic definitions that will be used throughout the work. Chapter 3 provides some insights into related works. Chapter 4 presents our ordering heuristics. Chapter 5 presents the libraries used to implement our approach and which also serve to measure baseline performance. Finally, Chapter 6 provides evaluation results for the described methods.

2. Preliminaries

This chapter contains definitions and examples of main concepts used in this work.

2.1. Boolean Formulas

This section contains definitions and notions including canonical normal forms of Boolean formulas.

Definition 2.1. Conjunctive Normal Form (CNF) is a conjunction of clauses $\bigwedge_i c_i$, where each clause c_i is a disjunction of literals $\bigvee_j l_j$. A CNF clause is *satisfied* if at least one of its literals is satisfied. A CNF formula is *satisfied* if all of its clauses are satisfied.

Definition 2.2. Disjunctive Normal Form (DNF) is a disjunction of terms $\bigwedge_i c_i$, where each term c_i is a conjunction of literals $\bigvee_j l_j$. A DNF term is *satisfied* if all of its literals are satisfied. A DNF formula is *satisfied* if at least one of its terms is satisfied.

Definition 2.3. An At-Most-One (AMO) constraint is a Boolean formula that takes a set of literals as an input and outputs *true* (is *satisfied*) if and only if maximal one of the input literals is satisfied. For a set of literals $\{l_1, \dots, l_n\}$ we use the following notation for the AMO constraint: $AMO(l_1, \dots, l_n)$

Definition 2.4. A Rich Conjunctive Normal Form (RCNF) formula is a conjunction of constraints, where a constraint can be a DNF formula, a disjunction of literals (equal to the CNF clause) or an AMO constraint. A RCNF formula is *satisfied* if all of its constraints are satisfied. Basically, RCNF is an extension of a CNF that allows more types of constraints, and thus allows smaller representation of a complex configuration rules .

2.2. The Boolean Satisfiability Problem

Let $\{x_1, \dots, x_k\}$ be a set of Boolean variables and q be a propositional logic formula in CNF that contains only literals of $\{x_1, \dots, x_k\}$. Formula q is *satisfiable* if and only if there exists a set of variable assignments, so that q is true. The Boolean Satisfiability Problem (SAT) is solved

if either the satisfying assignment of the formula is found or it is determined that the formula is not satisfiable.

2.3. Binary Decision Diagram

This section is based on the Handbook of Model Checking [3].

Definition 2.5. A Binary Decision Diagram (BDD) represents a Boolean function as an acyclic directed graph, with the nonterminal vertices labeled by Boolean variables and the leaf vertices labeled with the values 1 and 0. Each nonterminal vertex v has two outgoing edges: $hi(v)$, corresponding to the case where its variable has value 1, and $lo(v)$, corresponding to the case where its variable has value 0.

Definition 2.6. An Ordered Binary Decision Diagram (OBDD) is a BDD for which an additional ordering rule applies: for each nonterminal vertex v associated with variable x_i and a vertex $u \in \{lo(v), hi(v)\}$ associated with variable x_j , we must have $i < j$

An OBDD can be reduced by eliminating redundant nodes and merging terminal and duplicate nodes. The result of such reduction is the Reduced Ordered BDD (ROBDD) [3]. This reduction can be performed in the time linear in the size of the original graph [4].

ROBDDs serve as a canonical form for Boolean functions, meaning that, for a given variable ordering, every Boolean function has a unique representation as a ROBDD. The construction of a ROBDD is essential to keep the BDD as small as possible, as the complexity of most algorithms that utilize BDD is dependent on the number of nodes/length of paths in the tree. In this paper, every mention of BDD refers to ROBDD.

Given a BDD of a function, we can answer these and other questions related to a SAT problem for a given instance. The function is satisfiable, if it does have a terminal node labeled with value 1. We can find a random solution for the formula by traversing the diagram from root toward the "1" leaf. The complexity of such algorithm is $O(h)$, where h is the height of the BDD. We can count the number of solutions by traversing the BDD and counting the paths. The complexity is $O(n)$, where n is the number of nodes. [2, 3]

The common strategy for BDD frameworks is to divide an overall function into smaller functions and creating BDDs bottom-up. We start by creating BDDs for single literals, and then subsequently use the BDDs from previous steps to create new ones by applying operations like AND, OR, XOR. The generalization of these operations is called Apply algorithm. The algorithm creates a BDD that represents the given result of applying the operation between the formulas of input BDDs. The overall time complexity of an Apply operation is $O(N_u \times N_v)$, where

N_u and N_v are the number of vertices in BDDs where vertices u and v respectively are the root nodes of input trees.

A BDD requires a defined variable ordering that will be followed along all paths of a diagram. The size of a BDD depends heavily on the ordering of input variables. Some functions can rise in size from linear to exponential in the number of variables due to a bad ordering. However, the problem of finding an optimal variable ordering to construct a minimum-size BDD is proven to be NP-complete and some functions don't have an optimal ordering [5]. Thus, instead of computing an optimal variable ordering, it is a common approach to use heuristics to generate a good ordering and use it during BDD construction.

2.4. Sentential Decision Diagram

This section is based on the work of Darwiche (2011) [6].

Sentential Decision Diagram (SDD) is a more recent technique of representing of propositional knowledge bases. SDDs are a strict superset of OBDDs and are inspired by two discoveries: structured decomposability and strongly deterministic decompositions.

To explain SDDs, we first define the decomposition that is used to construct this type of decision diagram and we define the important notion of *vtrees*.

Definition 2.7. Consider a Boolean function $f(X, Y)$ with non-overlapping variables X and Y . If $f = (p_1(X) \wedge s_1(Y)) \vee \dots \vee (p_n(X) \wedge s_n(Y))$ then $\{(p_1, s_1), \dots, (p_n, s_n)\}$ is called a (X, Y) -decomposition of function f . We call each pair (p_i, s_i) an element of the decomposition, p_i a prime and a s_i sub. If $p_i \wedge p_j = \text{false}$ for $i \neq j$ the decomposition is called *strongly deterministic* on X .

Definition 2.8. A vtree for variables X is a full binary tree whose leaves are in one-to-one correspondence with the variables in X .

The vtree is used to recursively decompose a given Boolean function starting at the root of the tree. The left subtree of each node corresponds to the X variables, and the right subtree to Y variables of the (X, Y) -decomposition. The SDD representation is then based on a recursive application of the presented decomposition technique. The formal definition of this operation is as follows:

Definition 2.9. Notation: $\langle \cdot \rangle$ denotes a mapping from SDDs into Boolean functions. α is an SDD that respects vtree v if:

- $\alpha = \perp$ or $\alpha = \top$
- $\alpha = X$ or $\alpha = \neg X$ and v is a leaf with variable X
- $\alpha = \{(p_1, s_1), \dots, (p_n, s_n)\}$, v is an internal node, p_1, \dots, p_n are SDDs that respect the left subtrees of v , s_1, \dots, s_n are SDDs that respect the right subtrees of v , and $\langle p_1 \rangle, \dots, \langle p_n \rangle$ is a partition.

An SDD that consists of a constant or a literal is called *terminal*. Otherwise, it is called *decomposition*. SDDs are canonical, which means that for a given vtree, every Boolean function has a unique representation of an SDD [6]. SDDs are a strict superset over BDDs. The variable ordering of a BDD will then correspond to the total order of the vtree, which is defined as a sequence of variables obtained from the left-right traversal of the vtree [6].

BDD-trees are twofold exponential in treewidth, whereas SDDs are just exponential. The SDDs are also as tractable as BDDs, but are more succinct both in theory and in practice [7]. There exist some Boolean functions that can be represented with at least exponential BDD size and only polynomial SDD size [7].

3. Related Work

This section provides some insights into existing researches of BDDs and SDDs.

3.1. Product Configuration Using BDDs

The work of Hadzic et al. [8] presents BDDs as an efficient solution to the configuration problem. The authors also describe how they applied this method practically in the commercial software product *Configit*. They highlight that BDDs can be efficiently applied in industry use cases, since they have several advantages over commonly used search-based configurators, including faster response times, better scalability, and improved rule quality [8]. However, the research just mentions variable ordering methods to optimize BDDs, but does not provide any examples of efficient heuristics.

3.2. Static Variable Ordering

Static variable ordering techniques attempt to determine a near-optimal variable ordering before constructing the BDD based on prior analysis of the input function [9].

Many algorithms that were proven to be efficient are described in the work of Rice and Kulhari (2022) [9]. It includes straightforward approaches like Dependent Count, Variable Appending, Sub-Graph Complexity etc., as well as different metric optimization heuristic techniques.

One example of static variable ordering techniques is the MINCE (Min Cut Etc.) heuristic proposed by Aloul et al. (2004) [10]. Its main idea is to partition the variables into groups with minimal functional correlation between variables in separate groups by translating it into *balanced min-cut hypergraph partitioning* problem.[9].

The authors of the MINCE heuristic conjecture that their heuristic captures structural properties of Boolean functions arising from real-world applications [10].

3.3. Dynamic Variable Ordering

In contrast to static variable ordering techniques, the dynamic ordering techniques attempt to adjust the ordering online during the construction of the decision diagram [9]. The idea was presented by Rudell (1993) [11] based on the observation that swapping two adjacent variables of a BDD can be implemented without major changes to the Boolean function library API [3]. One of such techniques is *sifting*. Variables are moved up and down in the ordering, until the algorithm finds a location that leads to an acceptable number of total vertices. Evaluation results show that sifting improves the memory performance, but it is also a time-consuming process [3].

3.4. Top-Down SDD Compiler

Most SDD constructing algorithms work analog to the BDD construction that we presented earlier in this chapter: create a decision diagram from smaller decision diagrams. This process is usually referred to as *bottom-up* compilation.

The work of Oztok and Darwiche (2015) [12] describes a *top-down* compiler constructing SDDs from CNF formulas. The top-down compiler produces a subset of SDDs called Decision-SDDs. The compiler utilizes techniques from SAT solvers and model counting algorithms to decompose a formula. Results presented in [12] show that the top-down compiler is consistently more performant than the bottom-up compiler.

The miniC2D software package created by the University of California includes code for SDD compilation based on the idea of a top-down compiler from [12]. The program doesn't produce the SDDs itself, but its output can be transformed to the SDD in linear time.

3.5. Multivalued Decision Diagram

Multivalued Decision Diagram (MDD) is another structure that is also proved to be efficient in product configuration area. MDDs can be seen as a generalization of BDDs, where a function can work with more than binary (true/false) values. Research by Andersen et al. [13] provides an analysis of MDD usage in an interactive cost calculation task. The research also highlights the importance of variable ordering for both MDDs and BDDs. The evaluations even show that MDDs can perform better than BDDs for presented tasks. Nevertheless, most variable ordering heuristics are generally considered to be applicable to both BDDs and MDDs [9], so we could also apply the heuristics that we overview in our research to MDDs.

4. Analysis and Approach

This chapter describes different methods of ruleset pre-processing, variable ordering and BDD construction strategies. We will discuss the usage of existing state-of-the-art methods like FORCE as well as suggest some new algorithms.

4.1. Variable Reordering

This section explores various variable reordering heuristics and their algorithms aimed at improving BDD construction speed. The algorithms prioritize low ordering time, manageable implementation complexity, and effective variable ordering specifically for RCNF formulas.

We implemented and experimented with the following two different variable ordering heuristics: Variable Frequency (VF) and FORCE (F).

4.1.1. Variable Frequency

We propose the variable frequency (VF) as an easy to implement and efficient heuristic that produces reasonable orderings. The VF heuristic evaluates the variable's influence on the function output using the frequency metric. This metric counts either overall appearances of each variable or the number of constraints containing this variable. Subsequently, the algorithm sorts the list of variables using these values in descending order. The heuristic can be seen as a modification of the Dependent Count heuristic described in [9], but used mainly for a more general type of decision diagram called Multi-Valued Decision Diagrams (MDDs).

The intuition behind this heuristic is that more constrained variables are placed on a level closer to the root of the tree, which allows them to shorten the paths to the terminal nodes. However, the frequency metric does not consider the semantics of the formula and can lead to a false conclusion about variable influence.

The frequency counting takes linear time in the number of constraints in a formula $\Theta(|C|)$, considering that we have information about each constraint's contained variables. Sorting takes $\Theta(|V|\log|V|)$. Overall, the algorithm takes loglinear time in the number of variables $\Theta(|V|\log|V|)$.

4.1.2. FORCE Heuristic

The FORCE heuristic which is described in the paper by Aloul et al. (2003) [14]. FORCE is introduced as an alternative to MINCE, as described in Section 3.2, and comes with a simpler implementation and orders-of-magnitude increased speed, while providing competitive results with MINCE.

The algorithm is based on the same observation as the MINCE heuristic: Related variables in satisfiability

typically participate in the same CNF clauses [14], so the heuristic reorders Boolean variables to place "connected" variables close to each other. FORCE transforms the variable ordering problem into the linear placement problem. The vertices of a hypergraph correspond to variables and edges correspond to clauses. Since in our case, the RCNF is used, the clauses are replaced with constraints for all the definitions.

The FORCE algorithm uses the force-directed placement instead of a min-cut placement. The idea behind it is that interconnected objects (vertices of a hypergraph or variables in our case) experience forces analog to springs according to the Hooke's law. The algorithm computes these forces and displaces the vertices in the direction of the forces iteratively.

After an initial ordering is given, the center of gravity of each hyperedge e is defined the following way:

$$COG(e) = \left(\sum_{v \in e} l_v \right) / |e| \quad (2)$$

with l_v denoting the index of a vertex v in a current placement.

The new position l'_v is calculated with the following formula in which E_v is the set of all hyper-edges connected to the vertex v .

$$l'_v = \left(\sum_{e \in E_v} COG(e) \right) / |E_v| \quad (3)$$

Thereafter, the vertices are sorted according to the newly calculated positions. These iterations continue until a given metric of ordering stops improving. As proposed in the paper, the total variable span metric is used and the iterations stop after the metric doesn't decrease after given number n of iterations. Additionally, the iterations number is bounded by $c \cdot \log|V|$, where c is a constant.

The worst-case time of the algorithm is $O(|C| + |V| \log|V| \cdot \log|V|)$ [14], where C is the set of constraints, and we assume that the average degree of hyperedges and the average degree of vertices are limited by a constant.

4.2. Constraint Reordering

Another approach to reduce BDD construction time is by manipulating the ordering of constraints in an RCNF formula. By strategically grouping certain constraints together, the time required for combining smaller BDDs during construction can be decreased. This not only results in a smaller BDD but also reduces the time for subsequent operations. Constraint reordering, particularly when combined with dynamic reordering, can be highly efficient as it minimizes the number of nodes in intermediate results, thereby accelerating the sifting operation.

We implemented the following two different constraint ordering heuristics: Variable Frequency (VF-C) and Modified FORCE (M-FORCE).

4.2.1. Variable Frequency

We propose a concept of variable frequency ordering for constraints. The idea is to sort constraints according to the variables that they contain (similar to the VF ordering for variables). Specifically, it evaluates which variables are most influential in the ruleset and places the constraints that contain such variables at the beginning of a ruleset. Analog to the variable ordering with the same name, it evaluates the influence of the variables using frequency metric.

The proposed algorithm works the following way: we use the results of variable frequency ordering (Section 4.1.1). Then we analyze which variable in each constraint is the most frequent in the whole formula. If there are several variables with the same frequency, the one with the lowest index is taken. The constraint is then associated with this variable, and we sort the constraints according to the frequency of their associated variables.

It should be noted that the method induces a partition over the set of constraints based on the associated variables. The partition is specified in Equation 4. Let C be the set of all constraints, and $MFV(c)$ the associated variable of constraint c , i.e., the most frequent one.

$$P = \left\{ [c] \subseteq C \mid c' \in [c] \iff MFV(c) = MFV(c') \right\} \quad (4)$$

So, in addition to the most frequent variables being added to the overall BDD in the first iterations, this approach also groups the constraints with the same variables.

4.2.2. Modified FORCE

We present the heuristic that utilizes the idea of the FORCE variable ordering heuristic by applying it to the constraint ordering. It uses the concept of interconnected objects and placement by measuring their forces, but uses constraints as objects and redefines the interconnected objects as constraints having the same variables.

Basically, the algorithm is a modification of the FORCE variable ordering, the difference is in the types of objects that it takes as input. We build the hypergraph by using constraints as nodes and edges as sets of constraints that obtain the same variable. For a set of variables $|V|$, a set of constraints C and hypergraph edges E definition looks like this:

$$E = \left\{ e_v \subseteq C \mid v \in V, \forall c \in e_v : v \in c \right\} \quad (5)$$

We then use Formulas 2 and 3 and run the same algorithm to find a constraint placement that minimizes

the total span of the hypergraph (C, E) . The number of iterations is bounded by $c \cdot \log |C|$

As we can assume that every variable is used at least once, it applies $|E| = |V|$. So, the worst-case time of each iteration is $|V| + |C|$ (analog to FORCE, we assume that average degrees of vertices and hyperedges are bound by a constant). The sorting takes $\Theta(|C|\log|C|)$, so the worst-case performance of the algorithm is $O((|V| + |C|\log|C|)\log|C|)$.

The hypergraph construction is not as trivial as in the case of the original FORCE heuristic. With the usage of mapping from variables to their parent constraints and a mapping of constraints to the edges attached to them, the overall time complexity of the hypergraph construction is $O(|C| + |V|)$, assuming that the number of variables in a constraint and the number of constraints containing a certain variable are bound by a constant.

The algorithm can be modified by assigning different weights to each constraint based on their influence on the output. These weights can be used in equations, such as the center of gravity and position formulas, to prioritize the faster movement of more influential constraints.

4.3. Diagram Construction

Given a constraint ordering, there can be several construction strategies on how to use that ordering to construct a BDD. The Apply algorithm is used to recursively create BDD from smaller BDDs starting with just variables. The order in which the algorithm is applied affects the construction time of a BDD and can be changed either by constraint reordering, which we discussed in the previous section, or by construction strategies.

In this section, we will present two construction strategies for RCNF formulas. We mention the commonly used Depth-First strategy and present the Merge Strategy.

4.3.1. Depth-First Strategy

A common straightforward approach: we append a smaller BDD to the overall diagram as soon as it gets constructed. For a RCNF formula, the strategy creates a constraint, appends it to the overall BDD tree and moves on with construction of the next constraint.

Constraint construction time stays bound by constraint size, whereby the overall tree increases its size with each iteration, which slows down the appending of the next BDDs.

4.3.2. Merge Strategy

We present the Merge Strategy as an alternative to the Depth-First. This strategy tries to solve the problem by dividing this problem into smaller ones.

First, we merge the first two constraints of them together, then we merge the resulting BDD with another

BDD of two constraints. We subsequently continue merging the BDDs containing the same amounts of constraints, until the overall BDD is built. The construction order can be represented as a binary tree (Figure 1).

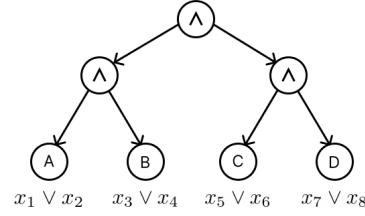


Figure 1: Merge construction tree

Example 4.1. Construction tree for a formula $(x_1 \vee x_2)(x_3 \vee x_4)(x_5 \vee x_6)(x_7 \vee x_8)$.

This way, constraint BDDs do not get appended in the exact order provided by this ordering, but global ordering is not influenced too much. For example, if we swap every two constraints (for instance, swap A and B, C and D in the example 4.1) the construction of their resulting tree will stay the same.

4.4. AMO Constraint Construction

An AMO constraint is not a binary operation, and its construction is not directly possible using frameworks like CUDD or libbdd that we will discuss later. Therefore, we have evaluated two ways on how to transform it into a form that uses Boolean operators.

The first way is to create a DNF representation of the AMO constraint, which is shown by Equation 6 and Equation 7 shows the whole formula f .

$$c_i = \left(\bigwedge_{j=1}^{i-1} \neg l_j \right) \wedge l_i \wedge \left(\bigwedge_{j=i+1}^n \neg l_j \right), \quad i \in \{1, \dots, n\} \quad (6)$$

$$f = \left(\bigvee_{i=1}^n c_i \right) \vee \left(\bigwedge_{j=1}^n \neg l_j \right) \quad (7)$$

In this case, the number of operations needed to build a BDD grows quadratically in the number of literals in the AMO formula.

Another way of presenting the AMO constraints is to use the XOR operation, which is also supported by the Apply algorithm. The formula constructed with XOR is shown by Equation 8:

$$\begin{aligned} f &= (l_1 \oplus l_2 \oplus \dots \oplus l_n \wedge c_{one}) \vee c_{zero} \\ c_{one} &= \bigvee_{i=0}^n \neg l_i, \quad c_{zero} = \bigwedge_{i=0}^n \neg l_i \end{aligned} \quad (8)$$

The number of operations grows linear in the number of literals, which makes it a more efficient method of building decision diagrams for AMO constraints.

4.4.1. SDD Vtrees and Variable Orders

As we discussed earlier, the SDDs variable ordering is more complex and is defined by vtrees instead of total variable ordering that is used in OBDDs.

Darwiche and Choi presented the following definition in [15]:

Definition 4.1. A vtree *dissects* a total variable order π if a left-right traversal of the vtree visits leaves (variables) in the same order as π .

In order to evaluate performance of the previously described BDD heuristics in the context of SDDs, we propose generating a total variable ordering, and then creating a vtree that dissects that ordering.

For one total variable ordering, there are many trees that can dissect it. Right-linear trees were discussed previously in section 2. SDDs that respect right-linear vtrees correspond precisely to the OBDDs, and therefore they cannot lead to any enhanced performance. Another choice are left-linear trees and balanced trees. The balanced trees were used for evaluation in [15] and are also supported by the framework presented in this paper.

5. Implementation

In this chapter, we describe the frameworks that allow constructing BDDs and SDDs using methods described in Chapters 2 and 4.

5.1. CUDD

CUDD¹ (Colorado University Decision Diagram) is an open-source state-of-the-art package for BDD manipulation written in C [16]. Practically, the package allows presents an implementation of the Apply algorithm and all needed data structures like unique table and cache.

The package also contains implementations of dynamic ordering algorithms. Available algorithms include sifting, window permutations, group sifting and others. The chosen algorithm will be used every time the number of nodes has increased up to a given threshold, which is set automatically after each reordering.

5.2. libssdd

libssdd is an open-source library for SDD construction and performing queries on them [17]. The interface and functionality of this package are very similar to the CUDD, but with respect to the SDD specifics.

¹<https://davidkebo.com/cudd>

We can create a vtree with a given total order and pass a parameter that specifies the type of the tree (right-linear, left-linear, balanced, vertical or random) that dissects a given total variable ordering. The library also allows automatic SDD minimization, which is similar to BDD dynamic ordering.

6. Evaluation

This chapter focuses on the evaluation of the algorithms described in the previous sections. We will take a look at the evaluated benchmarks and compare the results of different program configurations on these benchmarks.

For the evaluation, we use a GPU computer with 64GB RAM, Intel Core i9-9900K 3.60Ghz CPU and Nvidia GeForce RTX 2080 Ti GPU.

We used CUDD framework to implement the BDD construction and ordering and libssdd for SDD construction and ordering.

6.1. Benchmarks

One of the CAS Software applications is the Merlin CPQ configuration tool. It allows creating configuration rules using different complex relations between products and product parts. Fundamentally, the program has to solve the SAT problem for product configurations.

The evaluated benchmarks consist of real product configuration rules of different companies that use Merlin CPQ. Each benchmark corresponds to a company and contains rulesets that describe company products. Their rules were converted from the Merlin CPQ format into boolean formulas. Each ruleset is an independent RCNF formula saved as a DIMACS file. Table 1 shows the benchmarks and the number of variables and constraints in each of them.

6.2. Evaluation Goals

We evaluate the construction times as well as diagram sizes for the baseline approaches and different configurations of our approach. This includes several combinations of variants of variable ordering, constraint ordering, construction strategies, and an optional prior conversion of RCNF to CNF. We compare these configurations to the total construction time of SDDs using vtrees generated by the miniC2D top-down SDD compiler.

Configurations in the experiments are named using abbreviations. The construction strategy is specified only if it differs from the default depth-first strategy. All configurations utilize sifting dynamic ordering and XOR operations for AMO constraints, as shown in the Section 4.4.

Benchmark	#	Sum #vars	Median #vars	Sum #constraints	Median #constraints
vms	31	35258 (37985)	977 (1019)	15301 (27692)	295 (489)
campers	4	9713 (14543)	2430 (3546)	25592 (44386)	4754 (9121)
heating	9	24331 (42320)	2730 (4637)	73564 (237325)	7488 (26245)
forklifts	36	483226 (682785)	13637 (19166)	735319 (1972435)	19315 (53567)
printers	263	346017 (414854)	1309 (1459)	640915 (1488435)	1473 (2364)
boards	41	25600 (35857)	471 (630)	26596 (69165)	520 (1071)
trucks	50	1378761 (2466468)	27601 (46294)	6026775 (39474710)	118075 (736760)
plants	10	30638 (36922)	3589 (4398)	31119 (57871)	2592 (5266)
circuits	8	4486 (5194)	614 (694)	6349 (9869)	688 (1193)
Total	452	2338030 (3736928)		7581530 (43381888)	-

Table 1

Benchmark sizes. Numbers in brackets refer to the values of benchmarks converted to CNF

Abbreviations	Description
VF	Variable Frequency (variable ordering)
VF-C	Variable Frequency (constraint ordering)
FORCE (F)	FORCE (variable ordering)
M-FORCE (MF)	Modified-FORCE (constr. ordering)
MERGE (M)	"Merge" Construction Strategy
mC2D	miniC2D vtree Ordering
cnf	Input rulesets converted to CNF

Table 2

Abbreviations used for heuristics

6.3. Evaluation Methods

When constructing a decision diagram, choosing a suboptimal ordering and dealing with numerous constraints in a ruleset can result in construction times lasting several days. To manage this, we impose a time limit during the construction process and evaluate the algorithm's coverage. If the total construction time exceeds the limit, the process is stopped, and the ruleset is considered unconstructed. We assess the algorithms by comparing the number of constructed rulesets and analyzing various statistics for each algorithm. For example, we compare the ordering time for cases where ordering was completed within the time limit.

First, we evaluate different configurations with a 5-minute limit and then use the most performant ones for construction with a 1-hour limit.

6.4. Ordering Time

Figure 2 shows how many instances of the whole set of ruleset can be ordered in a time given by the y-axis. Here, we evaluate only individual heuristics (with an exception to M-FORCE/FORCE/MERGE/cnf), since the present variable and constraint ordering heuristics are independent, and configurations with both methods being used will just have an ordering time that equals the sum of variable and constraint orderings. In contrast, the CNF benchmark changes the ordering time, since the number of variables and clauses is higher (as can be seen from Table 1).

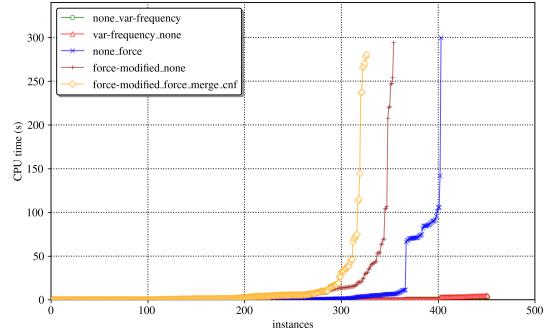


Figure 2: Time needed to find ordering using different configurations in 5 min limit

We can observe that the only configurations managing to order all instances under the limit are either VF or VF-C heuristics. The FORCE variable heuristic comes close to ordering all instances. M-FORCE constraint heuristic is even less performant, which can be explained with the fact that the number of constraints has more influence here and this number is bigger than the number of variables in the given benchmarks. The combination of M-FORCE and FORCE heuristics applied to the rulesets converted to CNF is the least performant of all, since the number of clauses in CNF rulesets is normally bigger than the number of constraints in original RCNF rulesets.

6.5. Coverage Statistics

In this section, we will evaluate the decision diagram construction with a time limit.

6.5.1. 5-Minute Limit

In Table 3 we can see the results for BDD construction with variable ordering, constraint ordering, and different construction strategies. trucks, heating and forklifts benchmarks did not result in successfully constructed rulesets and are therefore not present. Merge

Bench	R R	\emptyset \emptyset	VF-C VF	MF F M cnf	VF-C VF M	MF F	MF F M
vms	22	24	24	26	24	24	27
campers	2	2	2	2	2	2	2
printers	82	85	99	119	125	144	162
boards	15	34	25	32	34	33	36
plants	0	0	1	1	3	0	3
circuits	4	7	7	8	7	8	8
Total	125	152	158	188	195	211	238

Table 3

5-minute coverage results with different constraint orderings (R, \emptyset , VF-C, and MF), variable orderings (R, \emptyset , VF, and F), construction strategies (M), and optional conversion to cnf

Bench	SDD MF F M	BDD VF-C VF	SDD MF F	BDD VF-C VF M	BDD MF F	SDD mC2D cnf	BDD MF F M
plants	2	2	0	3	2	8	3
heating	0	0	3	0	0	0	0
circuits	7	7	0	8	8	7	8
trucks	0	0	8	0	0	0	0
vms	24	24	24	25	29	27	29
boards	26	31	29	35	36	33	36
printers	83	122	139	145	168	170	176
campers	2	2	2	2	2	2	2
Total	144	188	205	218	245	247	254

Table 5

Decision diagrams constructed in 1-hour limit

strategy consistently outperforms depth-first in every configuration. The best performance was achieved by the M-FORCE/FORCE/MERGE configuration, which also showed better results when the formula was given in RCNF rather than CNF.

Bench	\emptyset \emptyset cnf	MF F M	MF F	\emptyset \emptyset	mC2D cnf
vms	11	23	22	22	27
campers	2	2	2	2	2
printers	72	72	96	113	168
boards	17	21	25	27	33
plants	0	1	3	0	7
circuits	3	7	8	6	7
Total	105	126	156	170	244

Table 4

5-minute coverage results for SDDs with different constraint orderings (\emptyset , and MF), variable orderings (\emptyset , and F), construction strategies (M), and optional conversion to cnf, including the baseline mC2D

In Table 4 we can see that the heuristics do not work as well for SDDs as for BDDs. Configurations using variable and constraint heuristics do not improve coverage, suggesting unsuitability for constructing efficient vtrees for SDDs. The best result is shown by the construction using vtrees created by miniC2D tool. We can see that M-FORCE/FORCE/MERGE BDD construction (Table 3) almost reaches the performance of best SDD configuration. trucks, heating and forklifts benchmarks again did not yield constructed rulesets.

6.5.2. 1-Hour Limit

In Table 5 we can see coverage results for the 1-hour limit. The table contains both results for SDDs and BDDs

The M-FORCE/FORCE/MERGE configuration manages to outperform SDDs constructed with vtrees from miniC2d. M-FORCE/FORCE also shows comparable results.

We can see that even the best configuration shows only 56,1% done rulesets. Forklifts did not yield any constructed rulesets and is not present in the table and heating and trucks yielded just a few. From Table 1 we can see that these benchmarks are the biggest in terms of both variable number and constraint number, which leads to long ordering time. Still, big rulesets from heating and trucks are only constructed by MF/F, which manages to create the most optimal ordering.

We also evaluated the numbers of nodes for the diagrams constructed with the 1-hour limit. Here, we take only the subset of rulesets that are covered by each evaluated configuration.

Bench	SDD MF F M	BDD VF-C VF	SDD MF F	BDD VF-C VF M	BDD MF F	SDD mC2D cnf	BDD MF F M
plants	63464	570814	5559	78494	1112080	4745	390098
printers	2155	7573	1908	6935	4668	3238	4559
vms	3937	4759	2091	5302	3280	3050	6170
circuits	3050	8141	3130	9946	12993	4975	29231
boards	5552	11306	2879	12200	6024	4611	6599
campers	3202	15079	1899	16447	2177	2015	7472

Table 6

Median values of node counts for each configuration

In Table 6 we can see that SDD M-FORCE/FORCE configuration has the lowest median node count on 4 benchmarks out of 6. SDD configurations generally have better scores than BDD. The best performance BDD configuration is M-FORCE/FORCE.

6.6. Evaluation Summary

M-FORCE/FORCE/MERGE configuration delivers the best results for overall BDD construction time and outperforms all SDD configurations in 1-hour limit. However, it provides results that have one of the highest nodes counts. The presented configurations also improve construction time for SDDs in 1 hour limit, but still are inferior to

some BDD configurations. In opposition to BDD, M-FORCE/FORCE results in the smallest number of nodes for SDD.

Some benchmarks (trucks, heating, forklifts) could not be constructed within the limit due to complexity that can be observed from number of variables and constraints. Such big instances need more time to be compiled or more complex ordering heuristics to be applied.

Overall, with presented ordering heuristics, BDDs are much more efficient in modelling the rulesets and show promising results for use cases of knowledge compilation.

7. Conclusion

In this paper, we examined methods to enhance the construction time and size of BDDs for RCNF formulas. We presented variable ordering and constraint ordering methods that utilize the ideas of commonly used variable ordering methods. Furthermore, we considered different tree construction strategies. Additionally, we discussed the application of all described methods for SDD construction using vtrees.

We evaluated heuristics on RCNF benchmarks, assessing coverage in different time limits and determining the best results for each configuration, and found that Modified-FORCE and FORCE can greatly improve the BDD construction time. Furthermore, we applied the variable ordering heuristics to construct balanced vtrees for SDD construction, and results showed that FORCE and Modified-FORCE result in the best decision diagram size among all configurations.

References

- [1] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, L.-J. Hwang, Symbolic model checking: 1020 states and beyond, *Information and computation* 98 (1992) 142–170.
- [2] D. E. Knuth, *The Art of Computer Programming*, Volume 4, Fascicle 1B: Binary Decision Diagrams, Addison-Wesley Professional, 2009.
- [3] R. E. Bryant, *Binary Decision Diagrams*, Springer International Publishing, Cham, 2018, pp. 191–217. URL: https://doi.org/10.1007/978-3-319-10575-8_7. doi:10.1007/978-3-319-10575-8_7.
- [4] D. Sieling, I. Wegener, Reduction of obdds in linear time, *Inf. Process. Lett.* 48 (1993) 139–144.
- [5] Bryant, Graph-based algorithms for boolean function manipulation, *IEEE Transactions on Computers C-35* (1986) 677–691. doi:10.1109/TC.1986.1676819.
- [6] A. Darwiche, Sdd: A new canonical representation of propositional knowledge bases, in: *IJCAI*, 2011.
- [7] S. Bova, Sdds are exponentially more succinct than obdds, *CoRR* abs/1601.00501 (2016). URL: <http://arxiv.org/abs/1601.00501>. arXiv:1601.00501.
- [8] T. Hadzic, S. Subbarayan, R. M. Jensen, H. R. Andersen, J. Møller, H. Hulgaard, Fast backtrack-free product configuration using a precompiled solution space representation, in: *Proceedings from the International Conference on Economic, Technical and Organisational aspects of Product Configuration Systems*, Technical University of Denmark (DTU), 2004, pp. 133–140.
- [9] M. Rice, S. Kulhari, A survey of static variable ordering heuristics for efficient bdd/mdd construction (2008) 13.
- [10] F. Aloul, I. Markov, K. Sakallah, Mince: A static global variable-ordering heuristic for sat search and bdd manipulation, *JOURNAL OF UNIVERSAL COMPUTER SCIENCE* 10 (2004) 1562–1596.
- [11] R. L. Rudell, Dynamic variable ordering for ordered binary decision diagrams, *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)* (1993) 42–47.
- [12] U. Oztok, A. Darwiche, A top-down compiler for sentential decision diagrams, in: *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI’15*, AAAI Press, 2015, p. 3141–3148.
- [13] H. R. Andersen, T. Hadzic, D. Pisinger, Interactive cost configuration over decision diagrams, *Journal of Artificial Intelligence Research* 37 (2010) 99–139.
- [14] F. A. Aloul, I. L. Markov, K. A. Sakallah, Force: A fast and easy-to-implement variable-ordering heuristic, in: *Proceedings of the 13th ACM Great Lakes Symposium on VLSI, GLSVLSI ’03*, Association for Computing Machinery, New York, NY, USA, 2003, p. 116–119. URL: <https://doi.org/10.1145/764808.764839>. doi:10.1145/764808.764839.
- [15] A. Choi, A. Darwiche, Dynamic minimization of sentential decision diagrams, *Proceedings of the AAAI Conference on Artificial Intelligence* 27 (2013) 187–194. URL: <https://ojs.aaai.org/index.php/AAAI/article/view/8690>. doi:10.1609/aaai.v27i1.8690.
- [16] F. Somenzi, CUDD User’s Manual, 2005. URL: <http://web.mit.edu/sage/export/tmp/y/usr/share/doc/polybori/cudd/node3.html>.
- [17] A. Choi, A. Darwiche, SDD Advanced-User Manual Version 2.0, Automated Reasoning Group Computer Science Department University of California, 2018. URL: <http://reasoning.cs.ucla.edu/sdd/doc/sdd-advanced-manual.pdf>.

Product Variant Master in the Construction Industry

Irene Campo-Gay^{1,*}, Lars Hvam¹

¹Technical University of Denmark, Køppels Allé 404, 2800 Kgs. Lyngby, Denmark

Abstract

The architecture, engineering, and construction (AEC) industry is increasingly exploring the potential of mass customization and its impact on digitalization. However, developing digital tools can be challenging in terms of defining, delimiting, and structuring a construction product platform. To address this, a suitable information model is crucial to translate the information from the real world into a subset of data that a configurator can handle. This research aims to identify the common characteristics of construction product platforms to enhance their deployment into an information model, the so called product variant master (PVM) model. The study adopts a case methodology approach, typifying product platforms in three construction companies, and evaluates the applicability of the PVM model. Based on the findings, a systemic framework is proposed for depicting construction product platforms within the PVM model. The research concludes that by adopting this framework, the industry can streamline the modeling process, facilitate collaboration, and pave the way for effective digitalization in the AEC sector.

Keywords

AEC Industry, Configurator, Product Platform, Product Variant Master

1. Introduction

During the last decades, the architectural, engineering, and construction (AEC) industry has followed two main trends. In the 1950s and 1960s, it followed a mass production development, and later, in the early 1980s, it switched to an individual customization approach. Currently, the industry is embarking on a new strategy to exploit the best of both paradigms, uniqueness, and commonality in construction. However, there is still seldom research that can support the AEC industry in this new journey [1].

Adopting a mass customization strategy implies a major audit of a company's business model, and the critical activity revolves around a proper definition of a modularized product range [2]. An established tool in the manufacturing industry to describe a company's product range is the product variant master (PVM) model. The PVM model provides a rational and overall view of the product range's structure, including the product families and their variants [3].

Hence, adopting a mass customization approach could boost digitalization in the AEC sector, and the first step entails defining the product platform.

The topic of utilizing configurator methods in the AEC industry is not a novel concept. In fact, knowledge experts have employed the PVM model in limited construction projects and its application has also been documented [4, 5].

ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

✉ ircag@dtu.dk (I. Campo-Gay); lahv@dtu.dk (L. Hvam)
ID 0000-0002-8962-5386 (I. Campo-Gay); 0000-0002-7617-2971
(L. Hvam)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

However, the validity and suitability of the PVM model, originally designed for manufacturing products, in representing product platforms within the construction industry, have not been thoroughly examined. Therefore, our objective is to identify the shared characteristics specific to construction product platforms and establish a systematic framework for their representation using the PVM model. The adoption of this framework will facilitate collaboration between knowledge representation experts and domain experts in the construction industry, facilitating a deeper understanding of the rationale behind construction product platforms. Consequently, more robust, logical, and comprehensive representations of the models can be achieved, streamlining the modeling processes and enhancing insights into the product itself. This, in turn, enables the development of IT tools that were previously hindered by the challenge of representing the complex structures inherent in AEC products.

Based on this premise, we have formulated the following research questions:

RQ1- How can product platforms be generically portrayed in the AEC industry?

RQ2- How can the PVM model be used for systematic representation of AEC information?

RQ3- What are the key differences in the application of the PVM model between the construction industry and the manufacturing industry?

The remaining paper is structured as follows. Section 2 provides a comprehensive theoretical background on product platform development within IT systems, product modularity, and the product variant master model. In Section 3, the methodology used in the research is described. Section 4 presents the findings from the case studies, including the development of a generic systemic framework for construction product platforms, the ap-

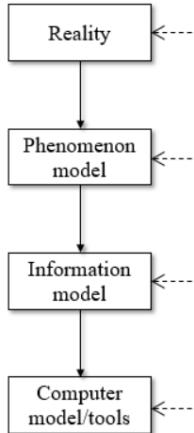


Figure 1: Process of translating the knowledge from the real word to an IT system. Adapted from Duffy et al. [6].

plication of the PVM model in the construction industry, and the differences in the application of the PVM in the construction industry compared to the manufacturing industry. Finally, Section 5 discusses the results and concludes the paper with implications for future research and practical applications.

2. Theoretical Background

2.1. Product Platforms in IT System Development

Defining product platforms is crucial in fostering mass customization and digitization in the AEC industry. In order to develop configurators that enable this level of customization, it is necessary to transform the knowledge of industry experts into a manageable subset of information [6]. The first step is the construction of a descriptive model that captures both explicit and tacit knowledge of the product. This knowledge is often dispersed across various departments within the organization. Such phenomenon model is collaboratively built with inputs from different domain experts and holds significant importance as it sets the foundation of the product platform architecture since it comprehensively defines the structure, functions, and properties of the product, encompassing its entire lifecycle. The next step involves formalizing the model to enable integration and modeling within an IT tool, such as a configurator. Formalization ensures that the knowledge represented in the phenomenon model can be effectively utilized in the development of a computer model tool. *Figure 1* illustrates this process.

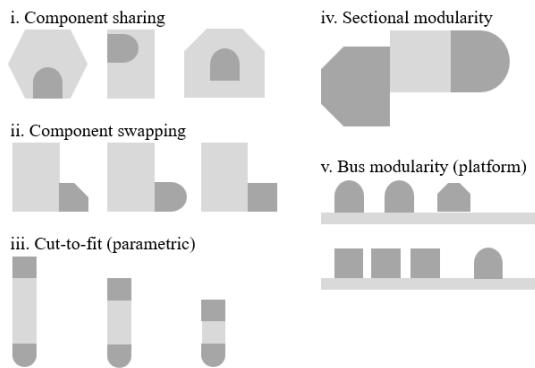


Figure 2: Modularity types based on [9, 10].

2.2. Product Modularity

One of the mass customization principles is modularization, which relates to using and arranging modules in a product architecture. There are many definitions of modularity and modules. However, one can describe a module as a definite object of a product with a distinct function and a defined interface to the other modules [2, 7]. The interface function is a crucial part of a modular product, and it should remain unchangeable as much as possible to grant the upgrade of modules over time [8].

The main types of modularity are depicted in *Figure 2* [9, 10].

1. *Component-sharing* modularity entails sharing modules across the product platform: E.g., the same engine used in different tools.
2. *Component swapping* modularity implies exchanging parts in a product: E.g., a phone with different case color options.
3. *Cut-to-fit* modularity concerns objects with parametric designs. E.g., a curtain cut with different lengths.
4. *Sectional* modularity involves the association without the restriction of modules: E.g., LEGO brick games.
5. *Bus* modularity (platform) means having the same interfaces for a base element. E.g., an Arduino board is a platform for electronic components.

2.3. Product Variant Master

A well-established modeling technique for developing product platforms is the PVM model. The PVM model provides a holistic view of a company's product platform.[2].

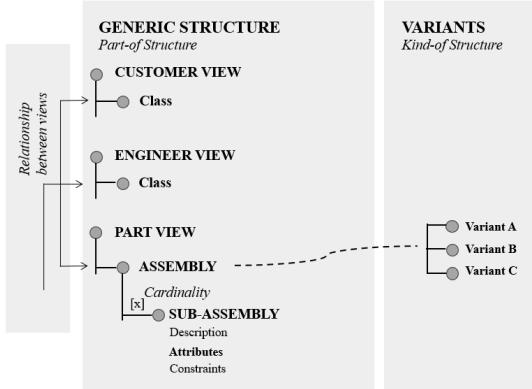


Figure 3: Basic notation of the PVM model.

The tool relies on three theoretical domains [3]. First, object-oriented modeling [11] makes it suitable for further developing digital tools. Second, the systems theory [12] provides the structure of the PVM. Third, modeling mechanical products [13], which is one of the reasons for this research to investigate the validity of using the PVM in the AEC industry.

The PVM technique, also named by some researchers as Product Family Master Plan (PFMP) [14, 15], provides a holistic, systemic representation of the information from three dimensions: the customer, the engineering, and the part view. First, the customer view reflects the customer's desire to buy the product. Second, the Engineering view contains the functions and principles to configure a solution. Third, the part view presents all the physical objects that can integrate the final product.

Moreover, the PVM is divided into two general sections. On the one hand, the left side of the PVM illustrates the generic structure or part-of structure with the different objects organized in a hierarchical structure. On the other hand, the right side of the PVM represents the variants or kind of structure, which describes the alternatives of the objects to the left.

Additionally, the generic structure is organized into classes further described by a cardinality property and a set of attributes and constraints. Finally, classes relate to instance connections on the left side of the PVM to represent when a class needs another class to fulfill its responsibility.

The PVM model is primarily used as a data collection method to retrieve information from the real world. Besides, it has a significant role as a communication tool to exchange and validate data with different knowledge experts. Building the PVM consists of multiple iterations that refine the model. *Figure 3* presents the basic notation of the PVM model.

3. Methodology

The case study methodology is a very suitable process in an exploratory investigation where research has yet not developed a theory. In this case, we opt for a multiple-case study approach to augment external validity. Nevertheless, we keep the number of cases to three to allow an in-depth analysis suitable for theory-building studies. Hence, we seek to achieve the generality of the conclusions while conceiving robust knowledge for the academic world [16, 17].

We developed and analyzed three different product platforms in three different companies. Our primary collection methods were semi-structured interviews, interaction with the various domain experts, and observations. On the other hand, we conducted data representation and documentation tasks mainly employing the PVM. Finally, we analyzed the information models under an iterative observation process of the PVM.

3.1. Case description

Companies 1 and 3 are medium enterprises with over 350 and 450 employees, respectively, while *Company 2* is a micro-enterprise with less than five employees. All companies operate in Scandinavian countries, Sweden and Denmark, and have embedded digital tools in their routine tasks to a certain extent, but only the third company has experience employing configurators. Moreover, each company performs in a different stage of the construction value chain and experiences a particular obstacle regarding a fragmented specification process. Table 1 provides an overview of the main distinctive features of the companies.

Company 1 pursues delivering more sustainable solutions to private investors to fulfill new governmental regulations. However, no digital tools can support them in developing environmental declarations, and they must resort to technical consultants to generate certified environmental declarations.

Company 2 seeks to speed the generation of quotes and bills of material to provide a faster response to private investors and agilely decide on the contractor by benchmarking.

Company 3 aims to speed up the design generation process. Even if they use digital tools to support different tasks during the process, no one can co-generate this design with the designers and potential customers and additionally include environmental assessment currently done in a separate operation.

All three companies have a shared approach when it comes to the configurator tool, which is seen as a decision support tool utilized by designers to adopt a proactive approach to design rather than a reactive one. This proactive approach helps prevent potentially high costs in sub-

Table 1
Features of the three analyzed company cases.

Features	Company 1	Company 2	Company 3
Employees	~ 350	~ 5	~ 450
Stage on the construction value chain	Construction materials	Main contractor	Construction materials
Product	Concrete products	Single-family houses	Façade Systems
Main collaborations with other parts of the construction value chain	Private investors (institutional) Architects Technical consultants on environmental declarations	Private investors (individual) Architects Technical consultants on energy assessment Technical consultants on structural assessment	Private investors (institutional) Architects Technical consultants on environmental assessment
Location	Sweden	Sweden	Denmark
Configurators experience	No	No	Yes
Organizational main problem	Boost more sustainable products	Reduce proposal lead times	Reduce design lead times
Main Construction Type	On-site and prefabricated	On-site	Prefabricated

sequent project phases. Additionally, in all cases, the configurator is integrated without the need for connecting external data or undergoing extensive reengineering processes. Therefore, the configurator successfully fulfills its primary objective of automating processes, relieving the workload on human resources, and speeding up lead times.

3.2. Data collection, representation, and analysis

We developed case studies related to *Companies 1* and *2* in parallel for 30 months until we produced functional and testable configuration system prototypes. On the other hand, we developed the case study in *Company 3* separately over seven months. In all cases, we gather the product information through modeling sessions with the

relevant domain experts in each case. The sessions were an hour long, and we held them mostly individually.

In *Company 1*, we had 35 sessions with the project leader, 24 sessions with an environmental assessor, and three sessions with the domain expert. Additionally, we held a testing workshop with the external project committee.

In *Company 2*, we held 115 sessions with the project leader and 36 sessions with technicians. Additionally, we evaluated the prototype with the potential users through a testing session followed by a semi-structured interview.

In *Company 3*, we held 20 sessions with the project leader and 20 sessions with the architectural firm in charge of developing the product platform design. We used open-ended questions to gather the data, and later, we reflected it in an ontology model, the PVM, which at the same time served as a communication tool with the domain experts.

Finally, we correlated the three PVM models through an observation analysis. Based on the discussions held in the research group, we developed the study findings under an iterative process to refine the results.

3.3. Research maturity

The results and findings presented in this paper are derived from an advanced stage of research. Due to confidentiality reasons, the specific PVM models utilized by each company cannot be disclosed. However, the subsequent sections describe the outcomes based on the aforementioned research.

Currently, both Company 1 and Company 2 have successfully adopted the PVM model, leading them to incorporate configuration systems into their work environments. These companies have integrated configurator tools using standard configuration systems as supplementary resources to alleviate the burden on human resources. In Company 1, the configurator tool is undergoing final validation, where engineers employ it to make more informed design choices. Similarly, Company 2 has reached a comparable stage, where the configurator replaces previously manual tasks, reducing lead time from weeks to hours. Importantly, these configurators do not interfere with additional software, such as CAD systems, as they are employed at different stages and outputs of the construction value chain. Additionally, Company 3 has also achieved success in developing a configurator tool, which has been operational for the past three years. This tool serves as a decision support resource for architects, providing assistance during early design phases of projects. In this case, the tool enhances early design phases of the project. It is worth highlighting that the PVM model played a strong role, drawing attention to various modular design components and assemblies on the platform that required redesign to facilitate the

subsequent development of the configurator.

4. Findings

We propose a generic framework to be used by AEC companies despite their stage in the construction value chain. For this purpose, we analyzed three product platforms in three companies with entirely different characteristics: company size, construction stage, product, digitalization aim, and on-site or prefabrication construction.

The main findings of the research are presented in the following three subsections. First, we describe the suggested systemic approach for developing product platforms in the AEC industries. Second, we illustrate how to use the PVM model in AEC projects to depict construction information. Finally, we highlight the main differences between the application of the PVM in the manufacturing industry compared to the AEC industry.

4.1. Systemic framework

Based on the analysis and observation from the three PVM models, we have identified a generic model applicable to any modular construction product platform embracing mass customization. The systemic framework comprises three layers: site, construction, and product.

1. This *site* depicts the place in which the construction is located. These can have relevance, for example, in terms of the transportation distance of the products from the factory to the working site, calculating the maximum structural load in the roof based on the average snowfall level, or knowing the accommodation capacity of construction machinery such as trucks or cranes, among other features. Moreover, the site layer can have more than one level, for instance, in renovation projects where both location and previous construction need to be considered.
2. The *construction* represents the volumetric shell in which the company's products are installed. In most cases, the *construction* might be broken up into *construction parts*. For example, the roof can be one of the *construction parts* of a building *construction*.
The predominant modularity type in this level is "cut-to-fit," which has the property of parametrization and, hence, describes the volumetric object.
3. The *products* layer illustrates the actual commercialized products. This layer is composed of multiple instances, and its total number depends on the project's complexity. There are two defined types of *products*:

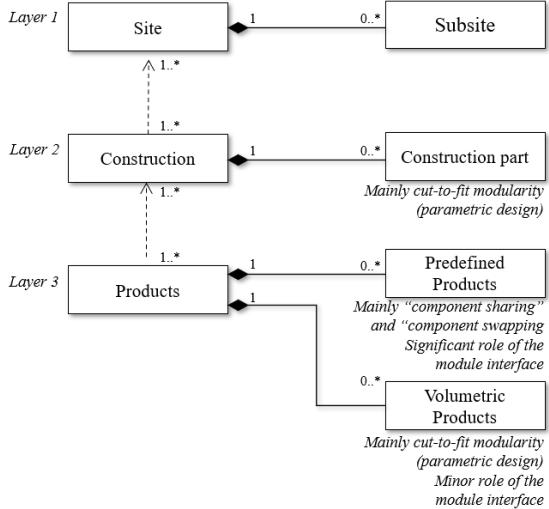


Figure 4: Systemic framework depicted using a UML diagram.

- a) *Predefined products* are predominant in prefabricated construction and are mainly defined by "component sharing" and "component swapping" modularity. The module interface is significant in predefined products and needs to be particularly well-defined. A frequent example of *predefined products* is windows and doors. Another example of a *predefined product* could be a modular room in which "cut-to-fit" modularity might also be present but in which "component sharing" and "component swapping" modularity have a more significant influence.
- b) *Volumetric products* are predominant in on-site construction, and they are mainly defined by "cut-to-fit" and "sectional" modularity. Hence, the module interface has limited significance, and its principal characteristic is its parametric design. An illustrative example of a *volumetric product* could be the concrete used to build a wall.

Figure 4 illustrates the generic systemic framework using UML notation.

4.2. PVM in the construction industry

The generic systemic framework facilitates the modeling process in the PVM model by providing a better understanding of the construction product platform. *Layer 1*, site, and *layer 2*, construction, are described in the *Customer View* since they directly depend on customer preferences and choices. Likewise, *layer 3*, products, is

depicted in the *Part View* as it represents all the physical components of the project. The three layers are closely related and utterly dependent on one another.

Figure 5 illustrates the applicability of the PVM information model in the construction industry. Besides, the generic systemic framework is reflected to envision its use in construction product platforms.

This reinterpretation of the PVM model can assist construction companies in portraying their product range, particularly in the early design phase of the information model. Hence, the PVM description could potentially reduce the time and resources invested in designing and organizing the modules and their relationship.

4.3. PVM application in industrial manufacturing vs AEC industry

Notable distinctions between the application of the PVM model in industrial manufacturing companies and its application in the construction industry have become evident. The following outlines the unique characteristics and novel approaches of the PVM model specifically tailored for the construction industry, in contrast to previous PVM applications, focused on mass-customized products in manufacturing:

- *Modularity*: Modularity in AEC projects relies heavily on design parametrization, i.e., cut-to-fit modularity.
- *Digitalization*: While configurators are widely employed by manufacturing companies to address mass customization, they are relatively unfamiliar tools in the AEC industry. Architects primarily rely on BIM (Building Information Modeling) tools, which are usually based on CAD systems lacking parametric history design capabilities, a crucial aspect considering the modular typification of construction projects.
- *Product structure*: The construction industry is distinguished by its offering of unique designs with a low degree of standardization in their building systems.
- *Stakeholder dependencies*: The construction value chain operates in more isolated silos compared to the manufacturing industry. There are substantial interdependencies among architects, engineers, and constructors, requiring extensive coordination efforts.
- *Production process*: Manufacturing processes are typically standardized and tightly controlled in a manufacturing environment. In contrast, construction projects predominantly involve on-site construction, encompassing numerous disciplines, manual operations, and coordination,

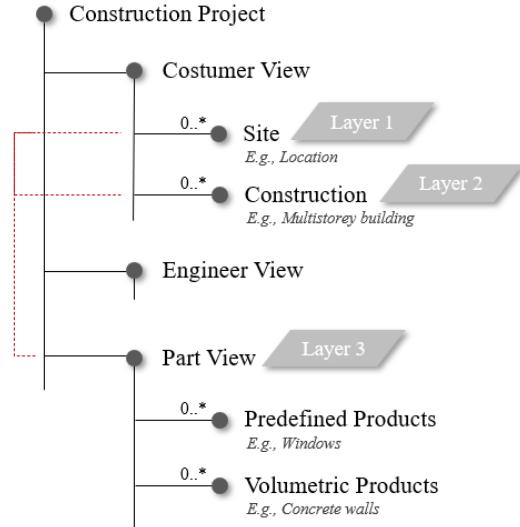


Figure 5: Applicability of the PVM information model in the AEC industry to depict construction product platforms.

which can present challenges in mapping out the production process.

- *Production variability*: Construction products exhibit higher tolerances compared to manufactured products, necessitating allowances and adaptations due to site-specific conditions and project-specific requirements.
- *Production volume*: Mass customized products in manufacturing companies usually target customization at higher volumes. Conversely, the construction industry typically operates at lower volumes and on a project basis.
- *Product life-cycle*: AEC industry products are primarily designed for long lifespans, and consequently, maintenance and renovation processes have a significant influence on the overall product.

These differences highlight the need for specialized approaches and considerations when applying the PVM model in the construction industry, acknowledging its unique characteristics and challenges.

5. Discussion and conclusion

In this paper, we conducted an analysis of construction product platforms and developed a systemic framework for their depiction using the PVM model. Although configuration project development methods have been used in the AEC industry, the suitability of the PVM model for representing construction product platforms has not been

thoroughly studied. Previous literature shows limited application of the PVM model in construction projects and, moreover, it was originally designed for industrial mechanical products. Therefore, our study aims to analyze the validity and applicability of the PVM model rather than its feasibility for construction product platforms.

Our research has three main contributions and outcomes:

Firstly, we developed a generic framework that provides a systematic organization of construction product platforms into modules. This framework characterizes the relationship and cardinality of these modules, describing them based on their modularity and interface significance. Implementing this framework can enhance collaboration between knowledge representation experts and domain experts in the construction industry, leading to a better understanding of construction product platforms. Consequently, more robust, logical, and comprehensive models can be created, streamlining the modeling processes and providing deeper insights into the products themselves. Additionally, this development of IT tools, which was previously hindered by the challenge of representing complex structures in AEC products, can be considerably improved. This framework also addresses RQ 1.

Secondly, the framework helps answer RQ 2 by demonstrating the applicability of the PVM model in AEC cases. Despite being initially designed for industrial manufacturing projects, our observations confirm its suitability in the construction sector. Thus, we can describe the use of the PVM model in the construction industry and validate its applicability beyond manufacturing projects.

Thirdly, we address RQ 3 by uncovering that the application of the PVM model in the construction industry diverges from its usage in industrial manufacturing. The construction industry has different characteristics, including modularity, digitalization, stakeholder dependencies, production processes, variability, production volume, and product life-cycle. Consequently, applying the PVM model in the construction industry requires specialized approaches. It is essential to recognize these differences to effectively use the PVM model in the construction industry.

To conduct a comprehensive analysis of the research questions, we deliberately chose a smaller sample size. The observed variations among the three company cases provide further evidence supporting the generalizability of our findings. The validity of the research outcomes is reinforced by the advanced stage of development of the configuration tools. However, in order to strengthen the framework even further, it would be recommended to replicate and evaluate the proposed framework in additional cases.

Furthermore, it is important to emphasize that widespread use of the PVM model in the AEC indus-

try can help streamline the fragmented value chain of construction projects, which often rely on siloed specification processes. Documenting construction product platforms using the PVM model can bring similar benefits to those achieved by manufacturing industries, such as easier maintainability and smoother development of the product platform. Additionally, this approach has the potential to reduce the modeling phase of the configurator. It is conceivable that other business fields beyond manufacturing or the AEC industry could benefit from the same rationale applied in this research. Therefore, further studies could contribute to the theory of information models, specifically investigating the applicability of the PVM model in fields such as logistics, services, or processes.

References

- [1] C. L. Thuesen, J. S. Jensen, S. C. Gottlieb, Making the long tail work: reflections on the development of the construction industry the past 25 years, *Proceedings 25th Annual Arcom Conference* (2009) 1111–20.
- [2] L. Hvam, N. H. Mortensen, J. Riis, *Product Customization*, Springer Publishing Company, 2008. doi:10.1007/978-3-540-71449-1.
- [3] N. H. Mortensen, L. Hvam, A. Haug, Modelling product families for product configuration systems with product variant master, *Ecai 2010* (2010).
- [4] A. Kudsk, L. Hvam, C. Thuesen, M. O. Grønvold, M. H. Olsen, Modularization in the construction industry using a top-down approach, *Open Construction and Building Technology Journal* 7 (2013) 88–98. doi:10.2174/1874836801307010088.
- [5] A. Kudsk, M. O. Grønvold, M. H. Olsen, L. Hvam, C. Thuesen, Stepwise modularization in the construction industry using a bottom-up approach, *Open Construction and Building Technology Journal* 7 (2013) 99–107. doi:10.2174/1874836801307010099.
- [6] A. Duffy, M. Andreasen, Enhancing the evolution of design science, in: *Proceedings of ICED*, volume 95, 1995, pp. 29–35.
- [7] J. K. Gershenson, G. J. Prasad, Y. Zhang, Product modularity: Definitions and benefits, *Journal of Engineering Design* 14 (2003) 295–313. doi:10.1080/0954482031000091068.
- [8] A. Ericsson, G. Erixon, Controlling design variants: modular product platforms, *Society of Manufacturing Engineers*, 1999.
- [9] B. Pine, *Mass customisation: The new frontier in business competition*, Harvard Business School Press, Boston, MA, USA (1993).
- [10] K. Ulrich, Fundamentals of product modularity,

- Management of Design (1994) 219–231. doi:10.1007/978-94-011-1390-8_12.
- [11] S. Bennett, S. McRobb, R. Farmer, Object-oriented systems analysis and design using UML, McGraw-Hill, 2006.
 - [12] L. Skyttner, General systems theory: Origin and hallmarks, *Kybernetes* 25 (1996) 16–22. doi:10.1108/03684929610126283.
 - [13] V. Hubka, E. Eder, Theory of technical systems : a total concept theory for engineering design (1988).
 - [14] A. Haug, Managing diagrammatic models with different perspectives on product information, *Journal of Intelligent Manufacturing* 21 (2010) 811–822. doi:10.1007/s10845-009-0257-y.
 - [15] U. Harlou, T. U. of Denmark, MEK, Developing product families based on architectures, Department of Mechanical Engineering, Technical University of Denmark, 2006.
 - [16] C. Voss, N. Tsikriktsis, M. Frohlich, Case research in operations management, *International Journal of Operations and Production Management* 22 (2002) 195–219. doi:10.1108/01443570210414329.
 - [17] C. Voss, M. Johnson, J. Godsell, Case research, in: Research methods for operations management, Routledge, 2016, pp. 181–213.