2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) | 979-8-3503-2263-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSE-COMPANION58688.2023.00020

# RM2DM: A Tool for Automatic Generation of OO Design Models from Requirements Models

Zhen Tian†, Yilong Yang†*, Sheng Cheng‡
†School of Software, Beihang University, Beijing, China
‡Software Engineering and Digitalization Center of China Manned Space Engineering, China
zhentian1@buaa.edu.cn, yilongyang@buaa.edu.cn, chengs@cmse.gov.cn

***Abstract*—Enterprise information systems focus on dealing with the complex business logic of collecting, filtering, processing, and distributing data to improve productivity and service in our daily lives. The successful development of enterprise information systems is a labor-intensive activity in software engineering, and it requires sophisticated human efforts for requirements validation and system design. Our previous work RM2PT can help to achieve a validated requirements model by automatically generating prototypes from requirements models to support incremental and rapid requirements validation. In this paper, we present a tool named RM2DM to further alleviate the problem of system development by supporting automatically generating a OO (Object-Oriented) design model of enterprise information system from the validated requirements model. We evaluate the tool through four case studies. The experimental result shows that all class diagram classes and 93.8% of sequence diagram messages can be correctly generated within 10 seconds. Overall, the results were satisfactory. The proposed approach can be further extended and applied for system development in the industry.**

**The tool can be downloaded at http://rm2pt.com/advs/rm2dm, and a demo video casting its features is at https://www.youtube.com/watch?v=lrs57CjzmU8**

***Index Terms*—Design, Design Model, Requirements, Model Transformation**

## I. Introduction

Enterprises use an enterprise information system to collect, transmit, store, maintain and display information. Small and medium-sized enterprises have extensive needs to establish customized enterprise information systems to meet their business requirements. The primary business function of enterprise information systems is data management, and many of these systems are developed using frameworks like JavaEE and SpringBoot. All of the frameworks are based on the anemic domain model. Since monolithic enterprise information systems have system architecture and design principles, we can extract them from design experience as transformation rules and automatically generate designs.

Software design is a core of the software development process, including two main stages: architecture design and detailed module design [1]. Excellent software design is the key to producing reliable and extensible software, which highly depends on the software development experience of software engineers [2]. State-of-art tools are dedicated to generating designs from requirements automatically to save labor and time and provide designers with design solutions for reference.

Our previous work RM2PT [3]–[6] provides features for requirements modeling, analysis, and automated prototype generation from requirements models. Users can rapidly validate and evolve functional requirements through RM2PT to produce high-quality validated requirements models. The **validated requirements model** [7]–[9]consists of 1) a conceptual class diagram, 2) a use case diagram, 3) system sequence diagrams for use cases, and 4) the formal contracts of their system operations in OCL (Object Constraint Language). Formal requirements [10] [11]can eliminate ambiguity, reduce the communication cost between designers and requirement analysts, and avoid design mistakes caused by unclear requirements descriptions. RM2PT works only in the requirements phase, but the resulting validated requirements model can be used as input to the design phase.

In this paper, we present RM2DM: a tool for automatic generation of OO design models from validated requirements models. RM2DM uses formal requirements models validated through RM2PT as input. The output **OO design models** consist of 1) a class diagram for the structure design and 2) sequence diagrams for behavior design. The class diagram is generated from the conceptual class diagram, which describes how the system operations are encapsulated into classes. The sequence diagrams are generated from system sequence diagrams that describe how the objects are collaborated to fulfill the system operation. The main features of RM2DM are as follows:

1) *Automatic generation of the class diagram.* RM2DM can generate fabricated classes, decompose the system operations into primitive operations and encapsulate primitive operations into classes. The class diagram describes the design of the static structure of the system.
2) *Automatic generation of sequence diagrams.* RM2DM can generate sequence diagrams for each system operation based on system sequence diagrams and system operation contracts. Unlike system sequence diagrams that describe the interactions between users and the system, sequence diagrams illustrate the dynamic behavioral design inside the system in terms of how objects within the system collaborate to accomplish system operations.

The remainder of this paper is organized as follows: Section

*Corresponding author: Yilong Yang (yilongyang@buaa.edu.cn)

979-8-3503-2263-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICSE-Companion58688.2023.00020



Authorized licensed use limited to: UNIVERSIDAD DE GRANADA. Downloaded on January 17,2024 at 14:24:29 UTC from IEEE Xplore. Restrictions apply.

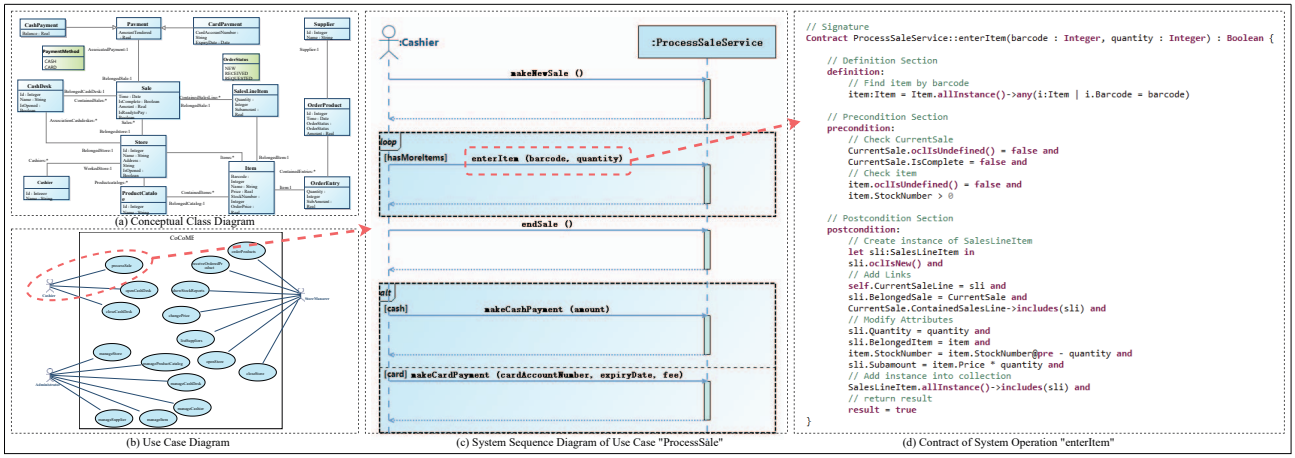



Fig. 1. Requirements Model

2 presents the overview of RM2DM. Section 3 introduces the features of RM2DM. Section 4 presents the evaluation results of the four case studies. Sections 5 and 6 discuss the related tools and conclude this paper.

## II. Overview

RM2DM provides a method to generate OO design models for enterprise information systems from requirements models automatically. In this section, we introduce the architecture of RM2DM and the anemic design model it is based. The architecture of RM2DM is shown in Figure 2. RM2DM takes validated requirements models as input and generates EMF[1] compliant OO design models.

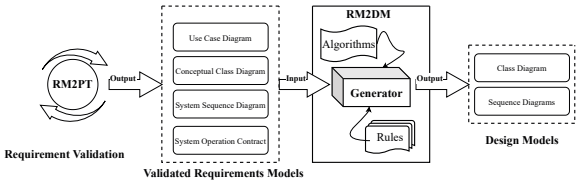



Fig. 2. RM2DM Overview

**The input requirements model:** The lightweight formal requirements model contains a use case diagram, a conceptual class diagram, system sequence diagrams, and contracts of system operations. During the requirements phase, users use RM2PT to rapidly generate prototypes for requirements validation and evolution, producing high-quality, validated requirements models. Figure 1 shows an example of requirements models.

**The output design model:** The design model consists of a class diagram and sequence diagrams. The class diagram describes the static structure and sequence diagrams are generated for each system operation, describing the dynamic behavior inside the system. Figure 3 shows an example of a class diagram, and a sequence diagram is shown in Figure 4.

RM2DM generates designs based on the design philosophy of the anemic domain model[2]. In the anemic domain model, entities only consist of data structures and primitive operations, and the business logic is in the service layer. The anemic domain model is a standardized design vastly supported by popular frameworks[3] like Java EE[4] and Spring[5] and is common in coding practice [12] [13], especially in web projects.

## III. RM2DM Features

In this section, we introduce the two features of RM2DM: generating the class diagram and generating sequence diagrams.

### A. Generation of Class Diagram

The generation of a class diagram involves three rules. RM2DM can generate fabricated classes, decompose the system operations into primitive operations and encapsulate primitive operations into classes.

*1) Preserve the Classes and Associations:* The first rule is to preserve the associations between classes and classes in the conceptual class diagram and add GET/SET methods corresponding to the class attributes. This rule ensures that the generated class diagram is consistent with the conceptual class diagram.

*2) Generate Service Classes:* The second rule is to encapsulate all use-case-related system operations into new service classes based on the use case diagram and the system sequence diagrams. The classes generated by this rule will serve as the service layer and provide services. The service classes initiate system operations, and then objects within the system collaborate to fulfill the system operations.

[1]https://www.eclipse.org/modeling/emf/

[2]https://martinfowler.com/bliki/AnemicDomainModel.html

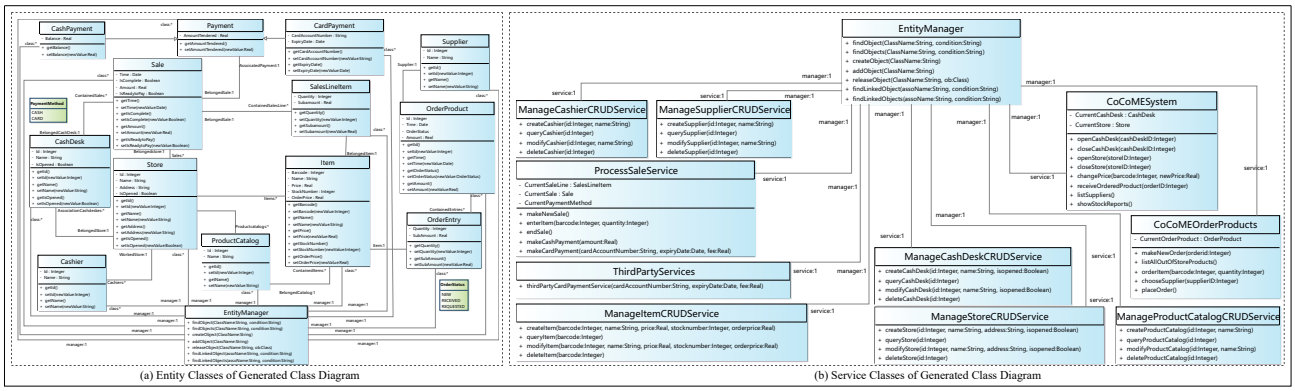
[3]https://www.continuum.be/en/blog/the-java-ecosystem-2022-survey-results/

[4]https://www.oracle.com/java/technologies/java-ee-glance.html

[5]https://spring.io/



Authorized licensed use limited to: UNIVERSIDAD DE GRANADA. Downloaded on January 17,2024 at 14:24:29 UTC from IEEE Xplore. Restrictions apply.




(a) Entity Classes of Generated Class Diagram (b) Service Classes of Generated Class Diagram

Fig. 3. Generated Class Diagram

*3) Generate Entity Manager Class:* The third rule is to create a new Entity Manager class associated with all other classes to add, delete, modify, and query objects. This class should therefore have primitive operations related to the addition, deletion, modification and query of objects.

### B. Generation of Sequence Diagrams

RM2DM uses sequence diagrams to describe the collaboration of objects within the system. The main content of the sequence diagram is the messages passed between objects. Based on the anemic domain model, there is a controller in the system. The controller sends the invocation messages of system operations to the corresponding service class. The service class sends messages to the related objects inside the system, and the objects collaborate to complete the system operations and return. The main problem is how we generate messages between objects. Of the three parts of the system operation contract, the OCL expressions in the precondition part describe the state the system should be in before executing the system operation. The preconditions do not involve interactions between class instances, so preconditions will not be converted into messages. The definition and post-condition of the system operation contract describe the system's dynamic behavior. The OCL expressions in definitions and post-conditions can be converted into messages in sequence diagrams. In order to transform OCL expressions in definitions and post-conditions into messages, a total of 17 transformation rules are specified. Transformation rules are presented in this form:

$$Rule : \frac{OCL\ Expression}{Message\ Label,\ Sender,\ Receiver}$$

The transformation rule contains two parts: the above is an OCL expression in the contracts, and the bottom part is the generated message and the message's sender and receiver. For each OCL expression in the system operation contract, apply rules 1-7 if it belongs to the definition part of the contract and rules 8-17 if it belongs to the post-condition part. Determine the rules for the OCL expression based on its syntactic structure. Apply the rules to extract the label, the sending object and receiving object of the message from the OCL expression.

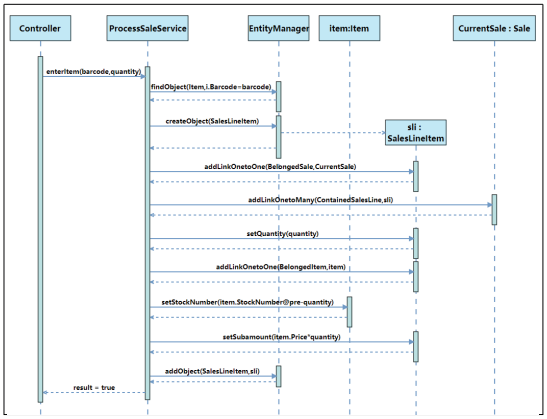



Fig. 4. Generated Sequence Diagram for System Operation "enterItem"

## IV. Evaluation

In this section, we present the case studies and then show the evaluation results based on the case studies.

### A. Case Studies

Four case studies from [3] are reused to evaluate RM2DM. These four cases describe the requirements of four common scenarios: Supermarket System (CoCoME), Automatic Teller Machine (ATM), Library Management System (LibMS), and Loan Processing System (LoanPS). The four requirements models contain 17 actors, 51 use cases, 137 system operations, 693 OCL expressions that are to be transformed, 39 entity classes, and 49 associations of entity classes. Detailed scenario descriptions and requirements models can be found on our website[6].

To illustrate the RM2DM's capabilities, CoCoME is used as an example to demonstrate the requirements model and the corresponding generated design model. The requirements model of CoCoME was built based on the description in [14], describing a supermarket trading system. This system has three

[6]https://rm2pt.com/casestudy



Authorized licensed use limited to: UNIVERSIDAD DE GRANADA. Downloaded on January 17,2024 at 14:24:29 UTC from IEEE Xplore. Restrictions apply.

roles: cashier, store manager and administrator. To process the sale, the cashier scans the products through the system operation enterItem and pays by credit card or cash after finishing scanning. Figure 1 shows the use case diagram and the conceptual class diagram of CoCoME, the system sequence diagram of Process Sale Service, and the system operation contract of system operation enterItem. Based on the use case diagram, conceptual class diagram, and system sequence diagrams, RM2DM generates the class diagram shown in Figure 3. According to the system operation contract, RM2DM generates the sequence diagram shown in Figure 4 to describe the collaboration of objects during the execution of system operation enterItem.

### B. Evaluation Results

The experimental settings of RM2DM are *3.5 GHz Intel Core i5, 16 GB DDR3, 500 GB Flash Storage, and JDK 11*. RM2DM can successfully and correctly generate all class diagrams of these four case studies. The results of the sequence diagram generation are shown in Table I. On average, 97.54% (DSRate) of the expressions in the definition part of the contract is correctly converted to messages in the sequence diagram, 92.99% (PSRate) in the post-condition part, and 93.80% (MSRate) of the overall messages are correctly generated. The result confirms the effectiveness of the tool. Generation for each case takes less than 10 seconds, showing that RM2DM is efficient in generating design models.

TABLE I
THE GENERATION RESULT OF SEQUENCE DIAGRAM MESSAGE

| Case Study | Def-OCL | Def-Message | DSRate | Post-OCL | Post-Message | PSRate | MSRate |
|---|---|---|---|---|---|---|---|
| ATM | 9 | 9 | 100.00% | 56 | 48 | 85.71% | 87.69% |
| CoCoME | 35 | 34 | 97.14% | 163 | 156 | 95.71% | 95.71% |
| LibMS | 59 | 57 | 96.61% | 244 | 223 | 91.39% | 92.41% |
| LoanPS | 19 | 19 | 100.00% | 108 | 104 | 96.30% | 96.85% |
| Sum | 122 | 119 | 97.54% | 571 | 531 | 92.99% | 93.80% |

* Def-OCL refers to the OCL expressions to be converted in the definition part of the system operation contract, and Def-Message refers to the number of messages correctly converted from Def-OCL; Post-OCL refers to the OCL expressions to be converted in the post-condition part of the system operation contract, and Post-Message refers to the number of messages correctly converted from Post-OCL.

## V. RELATED TOOLS

The tools similar to RM2DM can be divided into two categories based on the input. Tools in the first category generate design models from requirements containing natural language. A system named *RequirementElictor* [15] is proposed to generate class diagrams from natural language requirements. FDCT [16] uses typed and structured requirements specifications to reduce ambiguity and generate the class diagram to describe the functional architecture. *aToucan* [17] automatically generates analysis models consisting of class, sequence, and activity diagrams from requirement models containing natural language. Although *aToucan* can generate sequence diagrams and activity diagrams, their content is limited to human-system interactions and lacks designs to collaborate objects within the system. The second type of tool uses the formal requirements model as input. *RUT* [18] can generate class diagrams from virtual prototype models of multi-jointed robots. A formal metamodel specifies the virtual prototype model used by *RUT*. In addition, some commercial tools such as MagicDraw[7], Modelio[8] and Visual Paradigm[9] and existing plugins in IDE like IntelliJ IDEA[10] support the generation of class diagrams or sequence diagrams from code. Such tools can not provide developers with detailed design solutions generated from requirements and have a different envisaged usage scenario than RM2DM.

As is shown in Table II, The related tools perform well in generating class diagrams, including attributes of classes, associations between classes, and operations of classes. However, these tools can not generate the detailed design of the collaboration of objects inside the system, which is the innovation of RM2DM. Besides, RM2DM uses formal requirements models as input, avoiding the errors caused by the ambiguity of natural language. Finally, RM2DM generates designs for the specific monolithic enterprise information systems scenario based on the popular anemic domain model, so the generated designs are practical in software development.

TABLE II
CAPABILITIES OF RELATED TOOLS

| Related Tools | Structure Modeling | | | Behavior Modeling |
|---|---|---|---|---|
| | Class Attributes | Class Associations | Class Operations | Objects Collaboration |
| REQUIREMENTS ELICITOR | ✓ | × | ✓ | × |
| FDCT | × | ✓ | ✓ | × |
| aToucan | ✓ | ✓ | ✓ | × |
| RUT | ✓ | ✓ | ✓ | × |

## VI. CONCLUSION AND FUTURE WORK

This paper presents the RM2DM tool based on our proposed approach to automated design model generation from requirements models for enterprise information systems. Our previous work, RM2PT, can help to achieve a validated requirements model by automatically generating prototypes from requirements models. RM2DM generates OO design models from this validated requirements model, further alleviating the problem of software development. The generated design models can be used as references for designers and shorten the time-consuming design phase. Four case studies evaluated the capabilities of RM2DM. The experiment result is satisfactory and shows that the tool is effective and efficient.

In the future, We will consider combining the anemic model and domain model design philosophies to guide the automated generation of designs, thus addressing the automatic generation of design models for a broader range of systems. Also, we consider generating web services from design models based on our previous work [19]–[21].


## ACKNOWLEDGMENT

This work was supported by The National Key Research and Development Program of China (No.2021YFB2501301).


[7]https://www.magicdraw.com
[8]https://www.modelio.org
[9]https://www.visual-paradigm.com
[10]https://www.jetbrains.com/idea



Authorized licensed use limited to: UNIVERSIDAD DE GRANADA. Downloaded on January 17,2024 at 14:24:29 UTC from IEEE Xplore. Restrictions apply.

## REFERENCES


[1] A. Eden and R. Kazman, "Architecture, design, implementation," in *25th International Conference on Software Engineering, 2003. Proceedings.*, 2003, pp. 149–159.

[2] G. Samarthyam, G. Suryanarayana, T. Sharma, and S. Gupta, "Midas: A design quality assessment method for industrial software," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 911–920.

[3] Y. Yang, X. Li, Z. Liu, and W. Ke, "RM2PT: a tool for automated prototype generation from requirements model," in *ICSE (Companion Volume)*. IEEE / ACM, 2019, pp. 59–62.

[4] Y. Yang, W. Ke, and X. Li, "RM2PT: requirements validation through automatic prototyping," in *RE*. IEEE, 2019, pp. 484–485.

[5] Y. Yang, X. Li, W. Ke, and Z. Liu, "Automated prototype generation from formal requirements model," *IEEE Transactions on Reliability*, vol. 69, no. 2, pp. 632–656, 2020.

[6] T. Bao, J. Yang, Y. Yang, and Y. Yin, "RM2Doc: A tool for automatic generation of requirements documents from requirements models," in *ICSE-Companion*. ACM/IEEE, 2022, pp. 188–192.

[7] X. Li, Z. Liu, and J. He, "Formal and use-case driven requirement analysis in uml," in *25th Annual International Computer Software and Applications Conference. COMPSAC 2001*, 2001, pp. 215–224.

[8] Z. Liu, H. Jifeng, X. Li, and Y. Chen, "A relational model for formal object-oriented requirement analysis in uml," in *Formal Methods and Software Engineering*, J. S. Dong and J. Woodcock, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 641–664.

[9] Z. Chen, Z. Liu, A. P. Ravn, V. Stolz, and N. Zhan, "Refinement and verification in component-based model-driven design," *Science of Computer Programming*, vol. 74, no. 4, pp. 168–196, 2009, special Issue on the Grand Challenge. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642308000890

[10] N. Macedo, J. Brunel, D. Chemouil, A. Cunha, and D. Kuperberg, "Lightweight specification and analysis of dynamic systems with rich configurations," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 373–383. [Online]. Available: https://doi.org/10.1145/2950290.2950318

[11] S. Easterbrook, R. Lutz, R. Covington, J. Kelly, Y. Ampo, and D. Hamilton, "Experiences using lightweight formal methods for requirements modeling," *IEEE Transactions on Software Engineering*, vol. 24, no. 1, pp. 4–14, 1998.

[12] K. Cemus, T. Cerný, L. Matl, and M. J. Donahoo, "Aspect, rich, and anemic domain models in enterprise information systems," in *SOFSEM*, ser. Lecture Notes in Computer Science, vol. 9587. Springer, 2016, pp. 445–456.

[13] O. Nikiforova and K. Gusarovs, "Anemic domain model vs rich domain model to improve the two-hemisphere model-driven approach," *Appl. Comput. Syst.*, vol. 25, no. 1, pp. 51–56, 2020.

[14] C. Larman, *Applying UML and patterns: an introduction to object oriented analysis and design and interative development*. Pearson Education India, 2012.

[15] G. S. A. Mala and G. V. Uma, "Automatic construction of object oriented design models [UML diagrams] from natural language requirements specification," in *PRICAI*, ser. Lecture Notes in Computer Science, vol. 4099. Springer, 2006, pp. 1155–1159.

[16] V. S. Sharma, S. Sarkar, K. Verma, A. Panayappan, and A. Kass, "Extracting high-level functional design from software requirements," in *APSEC*. IEEE Computer Society, 2009, pp. 35–42.

[17] T. Yue, L. C. Briand, and Y. Labiche, "atoucan: An automated framework to derive UML analysis models from use case models," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 3, pp. 13:1–13:52, 2015.

[18] H. S. Son and R. Y. C. Kim, "Automatic transformation tools of UML design models from virtual prototypes of multi-jointed robots," *Multim. Tools Appl.*, vol. 77, no. 4, pp. 5083–5106, 2018.

[19] Y. Yang, W. Ke, W. Wang, and Y. Zhao, "Deep learning for web services classification," in *2019 IEEE International Conference on Web Services (ICWS)*, 2019, pp. 440–442.

[20] Y. Yang, N. Qamar, P. Liu, K. Grolinger, W. Wang, Z. Li, and Z. Liao, "Servenet: A deep neural network for web services classification," in *2020 IEEE International Conference on Web Services (ICWS)*, 2020, pp. 168–175.

[21] Y. Yang, Z. Li, J. Zhang, and Y. Chen, "Transfer learning for web services classification," in *2021 IEEE International Conference on Web Services (ICWS)*, 2021, pp. 185–191.




Authorized licensed use limited to: UNIVERSIDAD DE GRANADA. Downloaded on January 17,2024 at 14:24:29 UTC from IEEE Xplore. Restrictions apply.