# A Semantics Modeling Approach Supporting Property Verification based on Satisfiability Modulo Theories

Jingqi Chen
*School of Mechanical Engineering*
*Beijing Institute of Technology*
Beijing, China
3120190305@bit.edu.cn

Jinzhi Lu*
*EPFL SCI-STI-DK, Station 9,*
*CH1015 Lausanne,*
Switzerland
jinzhi.lu@epfl.ch

Guoxin Wang*
*School of Mechanical Engineering*
*Beijing Institute of Technology*
Beijing, China
wangguoxin@bit.edu.cn

Lei Feng
*KTH - Royal Institute of Technology*
Stockholm, Sweden
lfeng@kth.se

Dimitris Kiritsis
*EPFL SCI-STI-DK, Station 9,*
*CH1015 Lausanne,*
Switzerland
dimitris.kiritsis@epfl.ch

2022 IEEE International Systems Conference (SysCon) | 978-1-6654-3992-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/SysCon53536.2022.9773841

***Abstract*—Property verification in Model-based systems engineering (MBSE) supports the formalization of model properties and evaluates the constraints of model properties to select an optimal system architecture from alternatives for trade-off optimization. However, there is a lack of an integrated method that property verification enables to be applied in multi domain specific modeling languages, which is not conductive to the reuse of property verification for different architecture and may increase the learning and use cost. To solve the problem, a semantic approach combining a unified modeling method GOPPRRE modeling method with Satisfiability Modulo Theories (SMT) is proposed to realize property verification. The syntax of the multi-architecture modeling language KARMA based on the GOPPRRE modeling method is extended to realize property verification based on Satisfiability Modulo Theories, which enables the KARMA language to verify the models by evaluating the constraints which are defined based on the model properties. The proposed approach supports the evaluation of property constraints defined by different modeling languages for trade-off optimization in a unified language. The approach is evaluated by a case of optimizing the matching between workers and processes in a multi-architecture modeling tool *MetaGraph* which is developed based on KARMA. From the result, such approach enables to evaluate constraints consisting of properties and select an optimal scheme from the alternatives.**

***Index Terms*—Property verification, KARMA, GOPPRRE, SMT, MBSE**

## I. Introduction

With the development of complex systems, the system presents the characteristics of increasing complexity and uncertainty, which bring challenges to the system development [1]. The challenges focus on the reliability, efficiency and communications for system developments across domains. Model-based system engineering (MBSE) provides a solution to support complex system development through the concept, design, analysis and verification using model across the entire system life cycle [2].

Property verification is one of formal verification methods which applied in MBSE [3] [4]. Property verification formalizes and evaluates the constraints consisting of model properties for checking whether the system models remains valid [5]. Property verification can be applied in the architecture trade-off efficiently. When the alternatives with different properties exist, the property verification is able to evaluate properties of each probable alternative. Therefore, property verification supports eliminating the models with the properties of inappropriate value that do not meet the defined requirements so as to obtain the optimal model with the appropriate properties that meet all requirements. There are two advantages when using property verification to solve the trade-off problem based on MBSE [21]. Firstly, the models with quantitative information in MBSE can visualize all alternative architectures in a simple way instead of showing alternatives one by one in the mathematical way. Secondly, the trade-off optimization is from global view instead of analyzing all probable alternatives one by one.

However, when the property verification for trade-off optimization is applied, there are still three challenges existing. 1) there is a lack of unified methods to execute property verification for trade-off optimization in MBSE for the complex system architecture, which can be applied in multi domain modeling languages and support the verification across models. 2) there is a lack of unified method to realize the information interaction between models and the property verification, which enables the management of the information of models and the solving process. 3) the existing method for property verification in MBSE involves multi adding tools or modeling language, which is unfavourable for the usage of

property verification.

To solve the above problems, a semantic approach combining the GOPPRRE modeling method and Satisfiability Modulo Theories (SMT) is proposed to realize property verification [6] [7]. The semantics and syntax of multi-architecture language KARMA based on the GOPPRRE modeling method is extended to realize property verification, which enables KARMA to evaluate models. Based on the GOPPRRE modeling method, the property verification can evaluate the different models built by different modeling languages because the model elements in multiple domains can be formalized in a unified way. Combining with Satisfiability Modulo Theories, property verification is enabled to operate the formalized model elements in the unified modeling environment to construct the first-order logical constraints according to the requirement and realize the verification.

The main contributions of this paper are as follows: 1) a property verification approach based on a unified modeling method is proposed. This approach operates the properties of models to construct logical constraints of properties and evaluates the satisfiability of constraints for verification and optimization. 2) The syntax and semantics of property verification is defined based on the multi-architecture language KARMA and Satisfiability Modulo Theories. Finally, this approach is validated by the case study.

The remainder of this paper is organized as follows. Section II describes the related work of property verification in MBSE, and summarizes the key motivations of this paper. The Section III introduces a theoretical basis, formal expression and implementation of property verification. In the Section IV, the proposed approach is validated through the case of the trade-off optimization for matching workers and production processes. Finally, Section V shows conclusions.

## II. Related work

Property verification belonging to the formal verification formalizes and evaluates the constraints of the formal models. Formal verification refers to the analysis and verification of the system's characteristics described in formal specification to check whether the system meets the requirements defined by the designers [3]. Because of the rigorous mathematical foundation of formal specification, the formal verification is widely used to detect and ensure the correctness, integrity and reliability of the system. Many engineering researchers combine the formal verification with the MBSE models for logical analysis to ensure the reliability of the system. [8].

Property verification has been widely applied in MBSE. Since 2007, Martin Gogolla *et al.* has focused on how to extend the functions of Unified Modeling Language (UML) [9].Object Constraint Language (OCL) [10] is a textual language for formulating constraints and queries with a mathematical formalism, which can be used to construct and evaluate constraints of property. Martin Gogolla combines the key elements of UML and OCL for the logical evaluation of models. The relationships of model elements are transformed into the Boolean satisfiability problem (SAT) for verification. This integration has been applied in a UML modeling environment.

Another method was proposed by Brucker in 2008 [11]. He develops a platform for a formal proof that is constructed by combining UML and OCL, which is used to check whether there are contradictions between model constraints. Perez proposes an improved method combining UML and OCL in 2019 [12], that is to adopt a paradigm based on constraint logic programming to reason and explore the design space of UML models combined with OCL. This approach is applied to verify the correctness and the preciseness of models, and to explore more model candidates which meet the design requirements. In addition to the approaches combined with the formal solvers, Kolovos proposed EVL (Epsilon Validation Language) [13], which abstracts the syntax and semantics of OCL to construct its own property language for the evaluation of models, which mainly aims at testing the consistency of model transformation [14].

Through the analysis of the literature, there still remain some limitations in the property verification in MBSE. Compared with other studies, the key motivations of this paper are as follows:

- Define a property verification method that supports verification applied in multiple modeling languages: At present, most researches on property verification are developed based on one modeling language so that it is difficult to reuse the same method for other specific languages. This paper aims to propose a property verification method, which can support properties formalization and verification for the models of different modeling languages in a unified way.
- Define a unified modeling language supporting multi-architecture modeling, property formalization and verification based on unified modeling method and Satisfiability Modulo Theories: At present, the languages of using property verification and modeling are separated and the language of property verification involves many languages and relevant solvers, which are not conducive to users' understanding and learning. The disunity of modeling languages and languages for property verification does not lead to the transformation of data from models and the data of verified results. Therefore, the property verification should be developed to support modeling, property constraint formalism and evaluation.

## III. Using property verification for trade-off optimization based on a semantic modeling approach

This section formally defines the formalization of trade-off optimization based on property verification and explains the theoretical basis of unified modeling and property verification. Finally, an implementation process of property verification is defined.

### A. Formalization of trade-off optimization

The trade off is a decision-making process which including a set of techniques to evaluate the characteristics of each system in order to find the optimal solution. Property verification enables to evaluate constraints consisting of model elements in a formal way [20]. The process of trade-off optimization using property verification can be abstracted as the following.

$$\begin{aligned} A &: \{A_1, A_2 ... A_i, ... A_n\}; \\ C &: \{C_1, C_2 ... C_j, ... C_m\}; \\ O &: \{O_1, O_2 ... O_k, ... O_o\}; \end{aligned} \tag{1}$$

The equation 1 defines the space $A$ of alternatives including multi alternative architecture where the subscript of $A$ refers to the number label of alternative architecture. The capital $C$ refers to the set of constraints that the system architecture should satisfy, which are abstracted from the requirements of stakeholders. The capital $O$ refers to the set of optimizing targets according to the stakeholders.

To optimize the trade-off, the execution process begins with modeling the trade-off problem including the space $A$ of alternatives. When the models are finished, property verification is applied to formalize the model elements corresponding to the requirement and the system architecture into the constraints $C$ and optimizing targets $O$ and evaluate the alternatives according to the constraints and optimizing targets. When the evaluation from property verification is finished, the optimal solution is selected from the alternatives.

### B. Theoretical basis of modeling and property verification

In this paper, property verification is developed based on a multi-architecture modeling language KARMA combining with Satisfiability Modulo Theories. KARMA language is selected as the basis to extend its syntax for property verification which is designed based on a GOPPRRE meta-meta modeling method [15]. GOPPRRE meta-meta modeling method is selected as the modeling basis because GOPPRRE meta-meta modeling method supports the most modeling concepts and relational models among meta modeling methods, which is considered to be the most expressive meta-meta modeling method [16]. This method has a complete formal basis and can formalize meta models and models in different domains. Though the GOPPRRE meta-meta modeling method provide complete formal basis for modeling, the constraints in complex logical form can not be formalized according to the GOPPRRE only. Therefore, KARMA language based on GOPPRRE is extended with Satisfiability Modulo Theories(SMT) that refers to a logic theory that contains a series of axioms that can test whether the logical constraints based on mathematical theories is satisfied [17]. Its advantage is that it supports the formalization of the first-order logic expressions containing a variety of complex mathematical axioms, which leads to a wide expressiveness for constraints of model properties.

*1) GOPPRRE meta-meta modeling method:* GOPPRRE meta-meta modeling method extracts the similar features of different domain meta models and abstract them into six meta-meta models. The basic elements of GOPPRRE meta-meta modeling method are followings: Graph, Object, Property, Point, Relationship, Role, and the interrelationships among the six meta-meta models referring to the Extension. The specification of GOPPRRE is as following:

- Graph represents a collection of the Object and the Relationship;
- Object is an entity in a diagram such as the class concept in UML;
- Property refers to the attributes of the other five meta-meta models, data types of Property includes the arithmetic, Boolean, String, array and matrix;
- Point is a port of an Object;
- Relationship is a connection between Objects or different Points of Objects;
- Role is used to define connection rule that map to related Relationships.
- Extension defines the interrelationships among the six basic meta-meta models, such as Graph contains Object and Relationship, Object contains Point, Relationship contains Role, and the other five meta-meta models contain Property.

Based on the the basic meta-meta models, a modeling framework is developed as shown in Figure 1, which includes meta-meta models (M3), meta models (M2), models (M1), and views in the real world (M0).

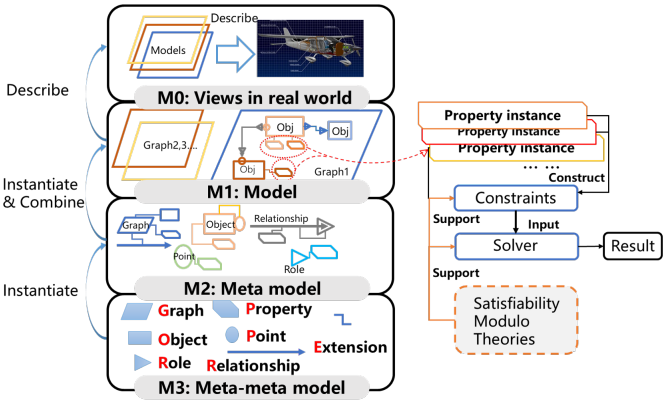



Fig. 1. GOPPRRE meta-meta modeling framework

M2 refers to the meta model, which is defined based on the meta-meta models and inherits the relationships among the meta-meta models. Meta models formally describe the components and development process of the system, which are used to construct models. M1 refers to the models, which are the combination of instantiated meta models. Combining the models to form a perspective of the real world is M0. The instance of model elements refers to the elements in M1 level.In this paper, the basic composition of property verification is the property instances in the M1 as shown in Figure 1, which is used to construct constraints.

*2) Satisfiability Modulo Theories:* Based on the GOPPRRE modeling method, multi-architecture language KARMA language can formally describe the model elements. However,

it can not further quantify and evaluate the alternatives. In order to support the description of property constraints and the implementation of property verification in MBSE, Satisfiability Modulo Theories is used as the theoretical basis to formalize the property instances to constraints of first - order logical expression and evaluate them in this paper. SMT is consisting of a series axioms based on first-order logical expression including multi mathematical theories. Moreover, SMT can not only provide the theoretical basis for checking the satisfiability of the first-order logic constraints, but also has corresponding solvers to implement the verification. When the constraints consisting of property instances are defined as first-order expression based on SMT, the constraints can be evaluated whether they are contradictory according to the axioms in SMT using the corresponding solver. Therefore, using SMT as the basis of extending KARMA language, the extended KARMA language is capable of formalizing and evaluating the constraints of property instances. Meanwhile, SMT and the corresponding solver enable the unification of the expression of properties and the process of solving the constraints of property instances.

### C. Property verification supporting trade-off optimization

Property verification is able to evaluate the constraints in models. Based on the GOPPRRE meta-meta modeling method and SMT, property verification is used to formally describe the property instances, to define the operators of property instances based on the first-order logical constraints, and to evaluate them in order to check whether there is a contradictory in the models. When the models are constructed according to the GOPPRRE modeling method, the property instances are captured from the models and used to define the logical constraints for implementing property verification.

Firstly, the alternatives $A$ defined in the formalization 1 are built as models based on GOPPRRE meta-meta modeling method. The alternatives are a group of different optional system architecture and each option may be the optimal one that meet all requirements. Such alternatives can be different in their structure such as the composition and connection, or the property instances or the above both. All kinds of alternatives can be abstracted into the constraints consisting of model elements, the alternatives consisting of different property instance is the point in this paper, the others will be considered in the future. For the alternatives that are different in the property instances, the value of property instances are different. Each value of property instance could be probable without any asserted limitations so that the optimal solution is not clear. The space $A$ of alternatives refers to the set of property instance with different value as Figure 2. The alternatives space is as the formalization 2.

$$\begin{aligned} A = \{ & \Sigma Property_{otherType}^{ProType}(otherID_{1i}, proID_{1i}), \\ & \Sigma Property_{otherType}^{ProType}(otherID_{2i}, proID_{2i}) ... \\ & \Sigma Property_{otherType}^{ProType}(otherID_{xi}, proID_{xi}) \}; \end{aligned} \tag{2}$$

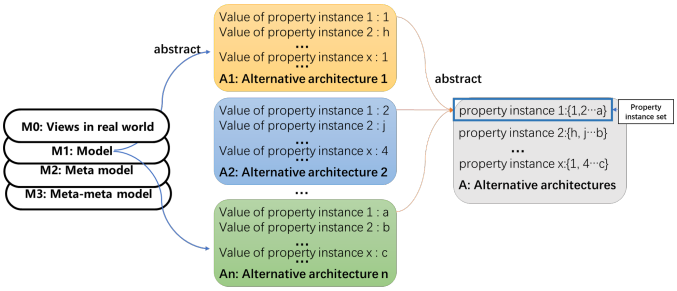



Fig. 2. The space of alternatives

The space $A$ of alternatives is the set of property instance sets including alternatives $A_1, A_2..A_n$. The $\Sigma$ refers to the set symbol which means the $\Sigma Property_{otherType}^{ProType}(otherID_{1i}, proID_{1i})$ is a set instead of a single instance. As the Figure 2 showing, each alternative is consisting of different property instances which are from the M1 model level. In the formalization of property instance, the subscript $otherType$ refers to the type of other five meta models and the superscript $ProType$ refers to the type of property instance such as String, Real, Integer. The IDs in the brackets refer to the name of some meta model and this property instance. The subscript of the IDs such as $1i$ refers to the number label of property instances. The first number $1$ refers to the property instance set and the number $i$ refers to the property instance in such set. The formalization 3 defines the property instance set which includes all possible property instances with different values.

$$\begin{aligned} & \Sigma Property_{otherType}^{ProType}(otherID_{1i}, proID_{1i}) = \\ & \{Property_{otherType}^{ProType}(otherID_{11}, proID_{11}), \\ & Property_{otherType}^{ProType}(otherID_{12}, proID_{12}), ..., \\ & Property_{otherType}^{ProType}(otherID_{1n}, proID_{1n})\}; \end{aligned} \tag{3}$$

The number $i$ of subscript $1i$ is replaced by $1, 2..n$ which refers to the label of property instance. Then, the space $A$ is defined while the optimal alternative is undetermined. The property verification can evaluate each probable property instance in the domain to eliminate the inappropriate ones that do not meet the defined requirements, and select the appropriate property instance to be determined.

When the optimal property instances in the property instance sets are required to be determined, property verification can generate a group of determined optimal property instances according to all the requirements and expected optimizing objective. The constraints are constructed by the captured property instances of the models and other elements, which has a return value of Boolean, through the mathematical operators of SMT as the formalization 4.

$$C_{Bool}^{j} = \odot(\Sigma Property_{otherType}^{ProType}(otherName, ProName), R). \tag{4}$$

$C_{Bool}$ refers to a unified description of constraints with property instances, the various operators and real number set

based on SMT which is extended from the $C$ defined in the formalization 1. $Bool$ refers that the return value of the expression is Boolean. The superscript $j$ presents the number label of constraints. $\odot$ refers to an operation process based on SMT that supports the mathematical operation based on SMT, including Boolean operators, arithmetic operators, array operators, comparison operators and some adding operators. The $\Sigma Property_{otherType}^{ProType}(otherName, ProName)$ refers to the set of property instances. The $R$ refers to the real number set. Generally, the and operation of defined constraints are applied to eliminate the inappropriate property instances as the following expression 5.

$$Constraint : C_{Bool}^{1} \wedge C_{Bool}^{2} .... \wedge C_{Bool}^{m} == true \quad (5)$$

A group of solution consisting of appropriate property instances is sifted according to the constraints based on the property verification, which satisfy all constraints without any contradictory. Then the optimizing target $O_k$, which combines the numeric operators and optimization operators, is formalized based on SMT. The optimizing target is specified as the following formalization 6:

$$O_{num}^{k} = M(\odot_{num}(\Sigma Property_{otherType}^{ProType}(otherName, ProName), R)). \quad (6)$$

$O_{num}^{k}$ is the objective that maximizes or minimizes an arithmetic expression, where the symbol $M$ refers to the optimization operator set, including minimization operator $Min()$ and maximization operator $Max()$. The subscript $num$ of $O$ presents the return value of optimized expression is numeric and the $k$ refers to the number label. For the optimized expression is numerical, the $\odot_{num}$ refers to the arithmetical operation based on SMT. When the property verification processed, a set of feasible solutions are finally output, that is, the property instance with optimal determined value.

### D. The semantic approach supporting property verification

This method combines the multi-architecture modeling language KARMA based on GOPPRRE meta-meta modeling method and SMT, so that KARMA is able to quantify properties and construct logical rationals among property instances. Then the KARMA language is used to describe the property constraints, and verify them. The implementation process is shown in Figure 3.

The process of property verification is implemented within *MetaGraph* [18]. *MetaGraph* is a software that being able to support meta-model development and modeling based on GOPPRRE modeling method using KARMA. The *MetaGraph* includes three main perspectives: modeling perspective, editor perspective and result perspective. When the engineers implement property verification, various models according to the design requirements are constructed in the modeling perspective of *MetaGraph* based on GOPPRRE modeling method firstly. Second, engineers make use of KARMA language to formally describe the constraints of property instances and the optimization targets according to their purposes. Third, *MetaGraph* compiles KARMA language, constructs and traverses the generated abstract syntax tree to execute the corresponding SMT solver to verify whether the constraints are satisfied. The result including whether the constraints are satisfied and the optimal solution that meets the given constraints automatically return to the result perspective which are displayed to the engineers.

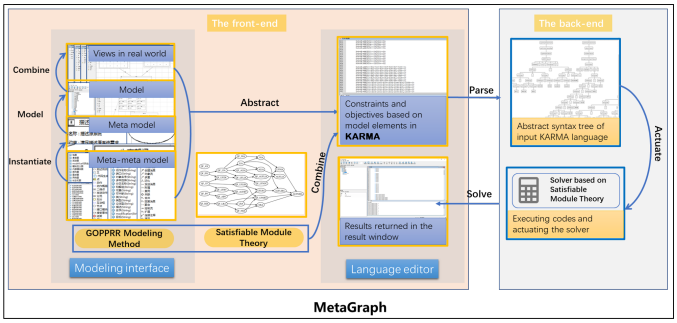



Fig. 3. Implementation of property verification

The abstract grammar tree of the extended KARMA language for property verification is shown in the Figure 4. The adding class $SMTAnalysisDeclare$ is used to formalize the property instances as the first-order logical expression and the $PropertyState$ is used to abstract property instances to be operated. The detailed description of the abstract class and the corresponding syntax are in the table I.

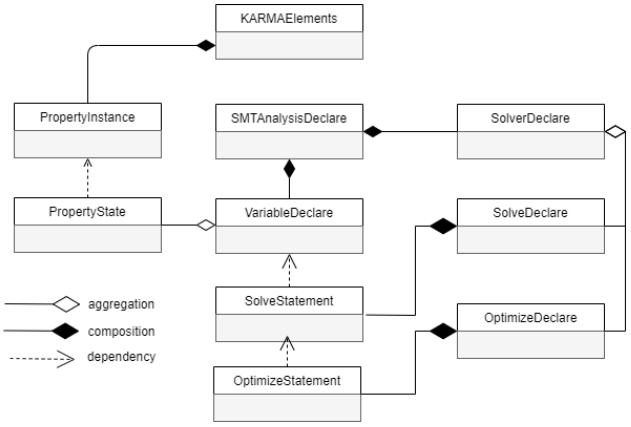



Fig. 4. Abstract grammar tree of the extended KARMA language for property verification

## IV. Case Study

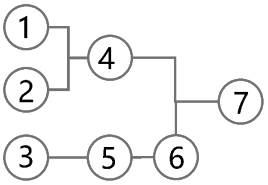
This section demonstrates a typical case to evaluate the validity of the property verification for trade-off optimization.In this case, the SysML modeling language is used to formalize the scheduling of production line, and the property verification is used to optimize the matching relationship between workers and processes of production line to obtain the most appropriate scheduling scheme.

TABLE I
THE SYNTAX OF KARMA LANGUAGE FOR PROPERTY VERIFICATION

| *Description* | *Syntax* |
|---|---|
| ***SMTAnalysis*** is used to declare the module of analysis for property verification | **SMTAnalysis** moduleName ..... **end** moduleName |
| ***PropertyState*** is used to abstract the value of property instance. | languageID. modelID. ObjectID. **Property**[PropertyID] |
| ***VariableDeclare*** is used to declare the variables including the Boolean, arithmetic, array, matrix, and String and the operation of variables. The value of property instance can be the composition of variables. | **Int** a;<br>**Int** b = **valueOf**(...);<br>**Int** maximum = max(a-b, b);<br>**Boolean** x = **true**;<br>**Boolean** y ;<br>**Boolean** z = x&& ($\sim$ *y*);<br>**Matrix** m = **IntegerMatrix**;<br>**initial**(3,3);<br>**Matrix** n = [1,1,1;1,1,1;1,1,1]+m |
| ***SolveDeclare*** is used to declare that the type of property verification is to evaluate the satisfiablity of constraints. | **Solve** solveName<br>...<br>**end** solveName |
| ***OptimizeDeclare*** is used to declare that the type of property verification is to optimize the variable value of property instances. | **Optimize** optimizeName<br>...<br>**end** optimizeName |
| ***solveStatement*** includes the statements of how to evaluate the constraints. | solveName. **add** (x && ($\sim$ y),...);<br>solveName.**push**;<br>%*construct a temporary stack*;<br>solveName. **pop**;<br>%*Pop up the temporary stack*;<br>solveName.**check**;<br>solverName.**solution**; |
| ***optimizeStatement*** includes the statements of how to optimize the variables. | optimizeName. **Add** (x&& ($\sim$ y),...);<br>optimizeName. **Push**;<br>optimizeName. **Pop**;<br>optimizeName.**Max**(a+b);<br>optimizeName.**Min**(a+b);<br>optimizeName.**AddSoft**(z,a)<br>optimizeName.**Check**;<br>optimizeName.**Solution** |

### *A. Property verification supporting trade-off optimization for matching relationship between workers and processes*

Optimizing the matching problems between workers and their working processes enables production line to improve work efficiency and avoid resource waste. There exists a assembly line with seven processes whose processing priority is shown in the Figure 5. Three workers are required to complete these seven processes. There are some known properties related to the processes: process operation time and process priority. There is no difference between workers' abilities which means that they can carry out any process with the same working efficiency. According to the known condition, the alternatives is shown as the Figure 6. If the models in SysML about matching problem are finished, the property instance which need to be determined is the relationship between each process and each worker. The value of such property instance can be 0 which means the worker does not execute the process or 1 which means the worker executes the process. The property instance set is consisting of the property instance with 1 and the property instance with 0 for different connections between workers and processes. The space $A$ for this case is consisting of a set of property instance sets corresponding to the relationship between workers and processes.




| Process | Time(s) | Process | Time(s) | Process | Time(s) |
|---|---|---|---|---|---|
| 1 | 37 | 4 | 62 | 7 | 55 |
| 2 | 16 | 5 | 29 | | |
| 3 | 57 | 6 | 31 | | |

Fig. 5. Process priority and the standard operation time of each process

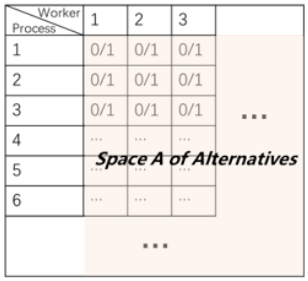



Fig. 6. Space of probable matching scheme

Firstly, SysML models are constructed based on the case to describe the requirements, background and solving method of the case shown as Figure 7. Figure 7.1 describes requirements for optimizing the matching relationships between workers and processes using Requirement Diagram. The requirements are defined from the perspective of stakeholders, including workers, customers and factory management departments. The main requirements are:

- Three workers must be assigned for at least one process and the workers do not change their working position.
- Each process must be carried out only once.
- When carrying out processes, the priority of the processes should be followed.

The optimizing target is to maximize work efficiency of production and minimize the working time of first worker for the working time of first station need to be limited because it is closed to obsolete.

The use cases involved in the process of production are described in Figure 7.2. After receiving the order from the customer, the production management formulates a production plan to arrange workers to produce on the assembly line in order. The processed products are delivered to the customer after meeting the requirement of ordered quantity. Figure 7.3 is an Activity Diagram, which describes the process sequence and corresponding processing time of the assembly line. The last Use Case Diagram illustrates the alternatives. The associated connection between the object instances workers and the object instances processes indicates the possibility between workers and processes in the Use Case Diagram. The value

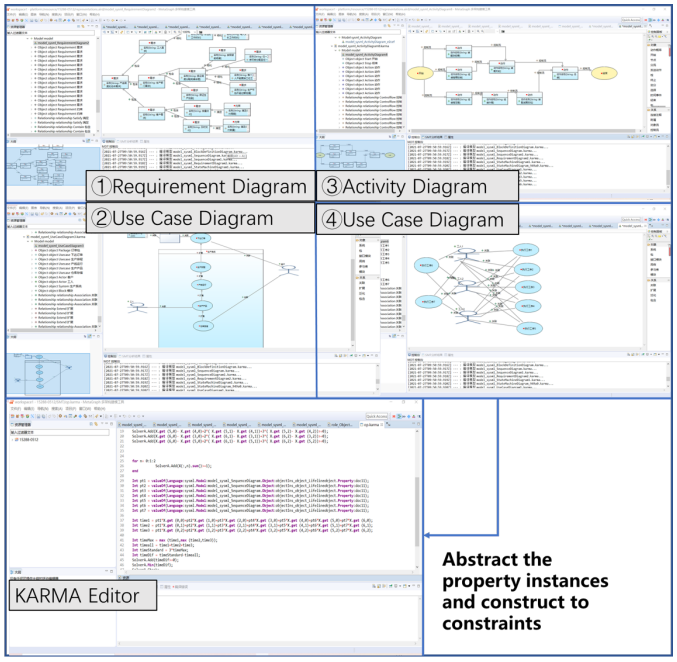



Fig. 7. Modeling for production scheduling in SysML

of property instance defined in the association connection between an worker and a process is variable including 0 or 1, which need to be determined as defined above.

In order to optimize the trade-off between efficiency and working time of first worker, the constraints and optimizing target are defined. Firstly, a 0-1 matrix $M_{7*3}$ consisting of property instances corresponding to the relationship between workers and processes is defined whose rows refer to processes and columns refer to workers. The first constraint is each process must be implemented.The second constraint is each worker must carry out one process at least and the constraint of processing priority. Then the optimizing target $O_{num}$ is defined as multi-objective expression with weights as the formalization 7.

$$\begin{aligned} O_{num} &= Min(\omega_t * T_1 + \omega_b * T_b) \\ T_b &= n \cdot T_{max} - \Sigma_{i=1}^{n} T_i \end{aligned} \tag{7}$$

The $Min$ refers to the minimizing operator. The minimizing expression is a multi-objective with weights $\omega$. The weight value can be altered according to the real situation. $T_1$ is the working time of the first worker. Efficiency can be expressed by balance delay time $T_b$ [19], which is defined in the second equation 7. $T_{max}$ refers to the maximum processing time among this 3 workers. If the balance delay time comes less, the waiting waste time is going to be less and the production efficiency becomes higher.

After the models constructed, the property instances are abstracted and formalized to the constraints in the bottom figure of Figure 7. When the constraints and optimizing objective are formalized in KARMA and the optimal scheme is generated, the result of the scheme returns to the result perspective as Figure 8 after 983 ms.

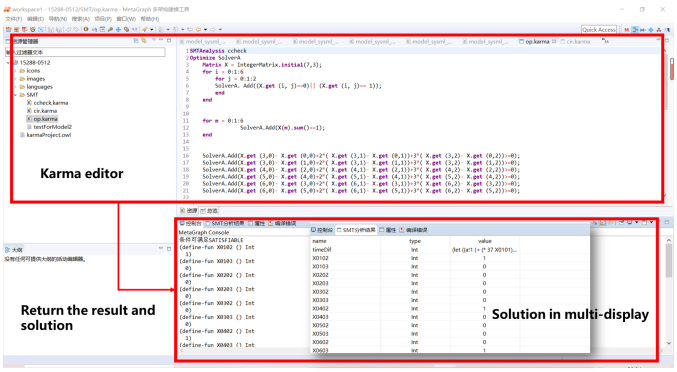



Fig. 8. Results and solution of matching relationship between workers and processes

When the weights are both 1, according to the calculation, the solution can be summarized as the following table II. The row refers to the number label of process and the column refers to the worker. The value of the cells shows whether the worker and the process are matched. If the value of the forth row and the second column of this table is 1, the worker2 carries out the process4. The balance delay time is 58s and the production efficiency is 83.79% and the working time of first worker is 57s.

TABLE II
SOLUTION OF THE MATCHING RELATIONSHIP BETWEEN WORKERS AND PROCESSES

| | Worker 1 | Worker 2 | Worker 3 |
|---|---|---|---|
| Process1 | 0 | 1 | 0 |
| Process2 | 0 | 1 | 0 |
| Process3 | 1 | 0 | 0 |
| Process4 | 0 | 1 | 0 |
| Process5 | 0 | 0 | 1 |
| Process6 | 0 | 0 | 1 |
| Process7 | 0 | 0 | 1 |

## B. Discussion

In this cases, the proposed property verification based on GOPPRRE modeling method and SMT is applied to formalize the property instances and operate the property instance as the first-order logical constraints to be evaluated. The multi-architecture modeling language KARMA with extended syntax and semantics based on SMT is used to be the unified language of modeling, formalization, operating the property instances and evaluation. Compared with other formal verification methods, property verification can support the formalization and evaluation of model elements in different domain languages based on GOPPRRE modeling method. At the same time, syntax of property verification is consistent with the way of model description (both are KARMA syntax), so as to ensure that the descriptions of model structure and property verification are consistent, and property verification can be carried out based on the models without conversion or other operations. However, there are still some limitations for the

current KARMA language. The current perform of property verification is in the limitations of the solver based on SMT. Its optimization solver does not perform well in supporting nonlinear optimization, which is easy to lead to long execution time and unstable results. Another limitation is that the trade-off optimization focus on alternatives with different property instance only in this paper. In the future, the other types of trade-off optimization will be considered.

## V. Conclusion

An approach to support property verification combining GOPPRRE meta-meta modeling method and SMT is proposed in this paper. Firstly, the relevant models are constructed based on GOPPRRE meta-meta modeling method. Secondly, the constraints of property instances are formalized and verified based on the SMT. Finally, through optimizing the matching relationship between the workers and processes, this approach of property verification is validated to evaluate the property instances and generate an optimal solution that meets the requirements. The validity of the proposed property verification to describe property constraints and evaluate them is illustrated. Based on the GOPPRRE meta-meta modeling method and the Satisfiability Modulo Theories, this method describes the models built in different modeling languages, and uses the solver under the unified Satisfiability Modulo Theories to complete the verification of constraints for trade-off optimization in a unifed way.

## VI. Acknowledgement

This work was supported by The National Key Research and Development Program of China (Grant No. 2020YFB1708100), the CN pre-study common technology (50923010101) and the Key RD foundation of Sichuan Province 2020YFG0352.

## References

[1] Hölttä-Otto, Katja, et al. "Design sprint for complex system architecture analysis." International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Vol. 51845. American Society of Mechanical Engineers, 2018.

[2] Friedenthal, Sanford, Regina Griego, and Mark Sampson. "INCOSE model based systems engineering (MBSE) initiative." INCOSE 2007 symposium. Vol. 11. 2007.

[3] Hasan, Osman, and Sofiene Tahar. "Formal verification methods." Encyclopedia of Information Science and Technology, Third Edition. IGI Global, 2015. 7162-7170. 271–350.

[4] Kerzhner, Aleksandr A., and Christiaan JJ Paredis. "Model-based system verification: A formal framework for relating analyses, requirements, and tests." International Conference on Model Driven Engineering Languages and Systems. Springer, Berlin, Heidelberg, 2010.

[5] Kolovos, Dimitrios S., Richard F. Paige, and Fiona AC Polack. "Eclipse development tools for epsilon." Eclipse Summit Europe, Eclipse Modeling Symposium. Vol. 20062. 2006.

[6] Wang, Hongwei, et al. "Ontology supporting model-based systems engineering based on a GOPPRR approach." World Conference on Information Systems and Technologies. Springer, Cham, 2019.

[7] De Moura, Leonardo, and Nikolaj Bjørner. "Satisfiability modulo theories: An appetizer." Brazilian Symposium on Formal Methods. Springer, Berlin, Heidelberg, 2009.

[8] Kern, Christoph, and Mark R. Greenstreet. "Formal verification in hardware design: a survey." ACM Transactions on Design Automation of Electronic Systems (TODAES) 4.2 (1999): 123-193.

[9] Gogolla, Martin, Fabian Büttner, and Mark Richters. "USE: A UML-based specification environment for validating UML and OCL." Science of Computer Programming 69.1-3 (2007): 27-34.

[10] Richters, Mark, and Martin Gogolla. "OCL: Syntax, semantics, and tools." Object Modeling with the OCL. Springer, Berlin, Heidelberg, 2002. 42-68.

[11] Brucker, Achim D., and Burkhart Wolff. "HOL-OCL: a formal proof environment for UML/OCL." International Conference on Fundamental Approaches to Software Engineering. Springer, Berlin, Heidelberg, 2008.

[12] Pérez, Beatriz, and Ivan Porres. "Reasoning about UML/OCL class diagrams using constraint logic programming and formula." Information Systems 81 (2019): 152-177.

[13] Kolovos, Dimitrios S., Richard F. Paige, and Fiona AC Polack. "On the evolution of OCL for capturing structural constraints in modelling languages." Rigorous Methods for Software Construction and Analysis. Springer, Berlin, Heidelberg, 2009. 204-218.

[14] Madani, Sina, Dimitrios S. Kolovos, and Richard F. Paige. "Parallel Model Validation with Epsilon." European Conference on Modelling Foundations and Applications. Springer, Cham, 2018.

[15] Kelly, Steven, Kalle Lyytinen, and Matti Rossi. "Metaedit+ a fully configurable multi-user and multi-tool case and came environment." International Conference on Advanced Information Systems Engineering. Springer, Berlin, Heidelberg, 1996.

[16] Kern, Heiko, Axel Hummel, and Stefan Kühne. "Towards a comparative analysis of meta-metamodels." Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, VMIL'11. 2011.

[17] De Moura, Leonardo, and Nikolaj Bjørner. "Satisfiability modulo theories: introduction and applications." Communications of the ACM 54.9 (2011): 69-77.

[18] Lu, Jinzhi, et al. "General Modeling Language to Support Model-based Systems Engineering Formalisms (Part 1)." INCOSE International Symposium. Vol. 30. No. 1. 2020.

[19] Kilbridge, Maurice, and Leon Wester. "The balance delay problem." Management science 8.1 (1961): 69-84.

[20] Leserf, Patrick, Pierre de Saqui-Sannes, and Jérôme Hugues. "Trade-off analysis for SysML models using decision points and CSPs." Software and Systems Modeling 18.6 (2019): 3265-3281.

[21] Parnell, Gregory S., et al. "MBSE Enabled Trade-Off Analyses." INCOSE International Symposium. Vol. 31. No. 1. 2021.