A Domain-specific Modeling Approach Supporting Tool-chain Development with Bayesian Network Models

Article *in* Integrated Computer Aided Engineering · November 2019

DOI: 10.3233/ICA-190612

5 authors, including:

Lu Jinzhi
Beihang University
126 PUBLICATIONS **1,322** CITATIONS
SEE PROFILE

Guoxin Wang
Beijing Institute of Technology
125 PUBLICATIONS **1,509** CITATIONS
SEE PROFILE

Xin Tao
Ericsson
11 PUBLICATIONS **164** CITATIONS
SEE PROFILE

Martin Törngren
KTH Royal Institute of Technology
248 PUBLICATIONS **5,490** CITATIONS
SEE PROFILE

A Domain-specific Modeling Approach Supporting Tool-chain Development with Bayesian Network Models

Jinzhi Lu[a] , Guoxin Wang[b,*] ,Xin Tao[a], Jian Wang[c,*] and Martin Törngren[a]
[a]*KTH Royal Institute of Technology, 10044, Stockholm, Sweden*
[b]*Beijing Institute of Technology, 100081 , Beijing, China*
[c]*University of Electronic Science and Technology of China, 611731, Chengdu, China*

Abstract. Constructing and evaluating a comprehensive tool-chain with commercial off-the-shelf and proprietary tools for the deployment of model-based systems engineering (MBSE) is a challenging and complex task. Specifically, the lack of early assessment during tool-chain development has led to increased research and development costs when unexpected features are developed or poor decisions are made. In this paper, a domain-specific modeling (DSM) approach is proposed to support decision-makings during tool-chain design and to facilitate quantitative assessment of tool-chain features at early-phases. Using this approach, different views of tool-chains are first formalized under a DSM framework. Then the DSM models are transformed to Bayesian network models for supporting the quantitative assessment of related tools in order to analyze the whole tool-chains' features. In the case study, the approach is verified by comparing two MBSE tool-chains for an auto-braking system design. The results indicate that the DSM approach enhances the understanding of tool-chain concepts, promotes the efficiency of MBSE tool-chain development, and verifies the tool-chain in early development phases using a quantitative approach.

Keywords: Bayesian network, Domain-specific modeling, Model-based systems engineering, Tool-chain assessment, Tool-chain formalism

1. Introduction

Tool-chains are becoming important to complex system development—especially that of aerospace equipment and aircraft—because they enable management of the evolution of aerospace systems, reducing cost and scheduling risks and promoting the efficiency of related product development [1]. The development of each complex system is unique; it is specific to domains, companies, and engineers. To support collaborative and concurrent development in different areas, model-based systems engineering (MBSE) has been proposed as a general approach for facilitating complex system development and integration. Using MBSE, tool-chains are proposed which are constructed by two or more modeling, simulation and design tools that, when combined, can support and construct a system engineering workflow. MBSE tool-chains constructed are expected to support the integration of multiple views of complex system development [2].

However, an MBSE tool-chain construction is challenging because many factors must be considered (e.g., tool features and interoperability between tools).

[*]Corresponding authors. Guoxin Wang, associate professor; 010-68912718; Room 343, Teaching Building I. No. 5 S. Zhongguancun Str. Haidian Dist. Beijing, China 100081; E-mail: wangguoxin@bit.edu.cn. Jian Wang, Associate professor; +86 13880639846; No.2006, Xiyuan Ave, West Hi-Tech Zone, 611731, Chengdu, Sichuan, P.R.China; E-mail: wangjian3630@uestc.edu.cn.

Often, tool-chain developers prefer to focus on technical aspects, such as tool mechanics, rather than truly understanding the system and modeling practices (what is the real need of system developers?) [3]. Moreover, tool-chain solutions must be prototyped and adopted to verify their functions which takes time and cost because of lack of quantitative analysis for MBSE tool-chain solutions. Without supports of a design approach, tool-chain developers also cannot obtain enough confidence and understandings of their products.

To address these challenges, a domain-specific modeling (DSM) approach is adopted to encompass design requirements, to plan and verify concepts of MBSE tool-chains. In this approach, developers formalize different views of the MBSE tool-chains, such as model workflows. Then, different tool selection rules are defined to provide various tool-chain construction solutions. In each tool-chain solution, specific tool assessment models are used to measure the **T**ool **M**BSE **C**apability **L**evel (TMCL, i.e., how does a tool support MBSE techniques?). Finally, developers employ DSM models to generate Bayesian network models for analyzing different tool-chain features and select a preferred solution.

The DSM approach aims to provide an efficient approach to designing MBSE tool-chain concepts, to analyze the tools' TMCL, and to provide guidance with which developers can make decisions about tool-chain development. The following research objectives in this study are explored:

- ***Support for formalizing MBSE tool-chains using a DSM approach***: DSM is used to formalize four views of tool-chains via the development of meta-models including model flows (to represent the relationships among models in the tool-chain), model tool selection views (to define tool-chain construction solutions), tool assessment views (to describe TMCL measurement) and tool-chain construction views (to describe tool-chain structures).
- ***Support for assessing TMCL using Bayesian networks***: The proposed TMCL measurement includes four aspects—application, tool-chain infrastructure, social procedure, and interoperability—each of which is supported by several defined metrics. The measurements are implemented based on Bayesian networks and provide a quantitative way to assess their tool-chains for early-stage tool-chain assessment and verification using a quantitative approach.

The remainder of this paper is organized as follows. Related work is introduced in Section 2. Then, the proposed approach is illustrated in Section 3. In order to verify this approach, two tool-chains are used to demonstrate how our approach supports MBSE tool-chain development in Section 4. Through comparing these two tool-chains, discussion and limitations are proposed in Section 5. Finally, our conclusion and future work are offered in Section 6. The acronyms included in this article are summarized in Table 1.

Table 1 List of Acronyms along with definitions

Acronym	Full definition
COTS	**C**ommercial **O**ff-**T**he-**S**helf
CPS	**C**yber-**P**hysical **S**ystem
CPTs	**C**onditional **P**robability **T**ables
DSM	**D**omain-**S**pecific **M**odeling
FMI	**F**unctional **M**ock-up **I**nterface
GOPPRR	**G**raph**, O**bject**, P**oint**, P**roperty**, R**elationship and **R**ole
MBSE	**M**odel-**B**ased **S**ystems **E**ngineering
LISI	**L**evel of **I**nformation **S**ystem **I**nteroperability
OSLC	**O**pen **S**ervices for **L**ifecycle **C**ollaboration
SPIT	**S**ocial **P**rocess **I**nformation and **T**echnique
SPIRIT	**S**ocial **P**rocess **I**nformation se**R**vice **I**nfrastructure and **T**echnique
TMCL	**T**ool **M**BSE **C**apability **L**evel

2. Related Work

MBSE tool-chains have been considered as one solution to support complex system development using systems engineering [1]. During MBSE tool-chain development, stakeholders need to consider different views in order to develop a tool-chain for their demands. For example, systems thinking is proposed for researchers to design their tool-chain concepts, because tool-chains are required by the whole lifecycle of complex systems [2]. Previous research has covered various aspects about tool-chain development: researchers made use of modeling approaches and designed measurements to formalize and to assess tool-chains in qualitative approaches. Moreover, quantitative assessment techniques are also adopted to measuring engineering systems which have be used to analyze tool-chains' features.

2.1. Formalisms for Supporting Tool-chain Development

Several researchers have theoretically investigated models to formalize general IT systems that support complex system development by referring to tool-chains. Cai et al. proposed three-layer ontology models for implementing automatic semantic annotation in 3D web scenes [4]. Shen et al. proposed ontology formalisms for sharing manufacturing scheduling and control information [5]. Lin et al. proposed a manufacturing system engineering ontology model for interenterprise collaboration [6]. These formalisms aimed to structure formal descriptions of related products and information. Zeid Kootbally et al. proposed an ontology-based approach for formalizing manufacturing kitting applications [7]. The ontology can improve the flexibility of the manufacturing configurations using robots. However, all the research contributed to their own domains which were all represented as ontologies aiming to support IT system operations.

Outside of IT system operations, several formalisms have been used for automating IT system development. Sabri et al. proposed an integrated semantic framework for developing an internet of robotic things (IoRT) [8], which provides an operational environment in which to build abstract models for generating semantic IoRT systems. Matthias et al. proposed a tool integration language for formalizing a service-oriented tool-chain [9]; the language representing tool-chains was used to generate codes in related tool adapters, aiming to support automated generation of open services for lifecycle collaboration (OSLC). H. Sun et al. designed ontology maintenance in order to realize high level architecture simulation automation [10]. Lu. et al. extended the tool integration language to support descriptions of co-simulation tool-chains [11]. These formalisms can all support IT system development, which mainly aims to generate relevant code for constructing IT systems.

2.2. Qualitative Assessments of Tool-chains

In order to assess tool-chains, several researchers have proposed standards for measuring performances of IT systems. Capability maturity model integration (CMMI) was standardized model for organization development [12]. CMMI represented the capability maturity of products (or IT systems) and provided one solution for improving these capabilities. The level of information system interoperability (LISI) is a standard for defining the interoperability of IT systems [13]. Technology readiness level (TRL) was also proposed for estimating the maturity level of critical technology elements during acquisition [14]. Although these standards defined different metrics and proposed solutions for supporting related measurements, most of these assessments were implemented manually and analyzed qualitatively.

Several researchers proposed solutions of qualitative assessments for tool-chain measurements. Dassisti et al. proposed an ontology-based model for assessing the interoperability of production-control systems [15]. Osterloh proposed a set of models supporting automated testing of related tools [16]. Bustillo et al. proposed a visualization technique using knowledge of industrial engineering for representing conditional inference trees. This visualization demonstrates the prediction data and models during machining processes to support engineers' decision-makings [17].

2.3. Quantitative Assessment Techniques Supporting Engineering Systems

Some researchers have used neural systems to model complex systems which can be used to support the system assessment. For example, L. Pan ([18]-[19]) adopted neural P System to model complex systems. Numerous studies have highlighted Bayesian networks for engineering monitoring, such as building structural health monitoring [20] and self-calibrating identification [21]. Bayesian network is also integrated with image processing to capture engineering features, such as image-based post-disaster inspection [22], analyzing workers 'behaviors [23] and feature extractions for motor imagery [24].

Moreover, Bayesian networks are used for making decisions in order to support the IT system development. Austin et al. proposed a Bayesian network model to assess TRL [25]. Eman et al. provided one learning cost-sensitive Bayesian networks for assessing credits for cyber security [26]. Schetinin et al. made use of Bayesian network models to support uncertainty estimations for air traffic conflicts [27]. Such developed models provided the realistic insights into predictive distributions for conflicts analysis.

Markov chain and Monte Carlo are other two methods techniques for supporting decision-makings. Tan Li et al. proposed a model-checking approach to predict the system risks in supply chain [28]. They

make use of Markov Decision Processes to model the stochastic behavior of supply chains. Pellegrinelli et al. make use of Markov Decision Processes to model human-robot interaction [29]. R. E. Banchs et al. make use of a Monte Carlo approach for sensor parameter estimation [30]. S. Donnay et al. proposed a design platform to support analog sensor interface architecture design using Monte Carlo [31].

The above-mentioned work all makes significant contributions toward supporting tool-chain formalisms and assessment. Moreover, Bayesian network models, Markov chain and Monte Carlo have been widely used to predict and simulate the performances of engineering systems. Compared with the target systems of existing contributions, MBSE tool-chains are also complex, even large-scale (if they need to support the entire lifecycle of the target system). They require related assessment approaches for early evaluation during tool-chain development, because reconstructing tool-chains leads to extra R&D costs. When tool-chains are developed for complex systems, the existing methods cannot support generalized tool-chain formalisms and assessments. Thus, a new method is required to support tool-chain development for formalizing and assessing MBSE tool-chains.

Moreover, most of the existing assessment techniques for tool-chains can only support qualitative analysis of their features which cannot provide confidential clues for decision-makings of tool-chain development. From the literature reviews, though Bayesian network models, Markov Chain and Monte Carlo were domain-specific purposes for each paper, but they already supported performance predictions for their own real-time applications. Currently, dynamic effects on tool-chains are not first considered before their development, because it is much important that tool-chain developers understand if the tool-chains satisfy their demands rather than dynamic efforts of the tool-chains. Thus, this paper focuses on Bayesian network to support qualitative analysis.

In summary, as emergent techniques aiming to support the entire system lifecycle, MBSE tool-chains are different from traditional tool-chains. First, these tool-chains need to support entire lifecycle of complex systems using systems engineering. All the design activities are supported by the tool-chains which mean tool-chain constructions are large-scale problems. Second, they must realize multi-domain models and tool integrations by considering interrelationships of tools. Third, different tool suppliers provide their own understanding of MBSE and related tools, leading to difficulties when tool-chain developers make decisions about tool selections. Therefore, the relevant formalisms of tool-chains are from different domains and hierarchical levels (e.g., system or component level). To address these challenges, a DSM approach is proposed to formalize tool-chain concepts and assess related commercial off-the-shelf (COTS) tools.

3. A Domain-specific Modeling Approach Supporting MBSE Tool-chain Development

In this section, an overview of the DSM approach is first introduced, and then a DSM framework using a GOPPRR (Group, Object, Port, Property, Role and Relationship) meta-meta models and Bayesian network theory.

3.1. Overview

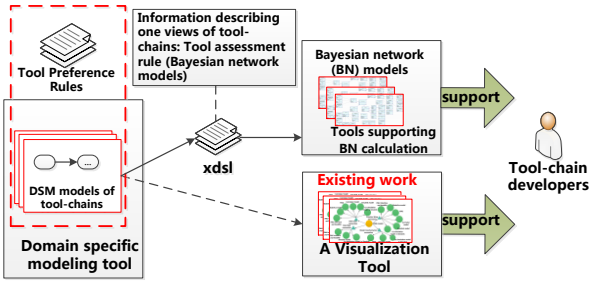

Fig. 1 The workflow of our approach

The Fig. 1 shows the workflow comprising our approach. A DSM tool is adopted to develop meta-models (introduced in Section 3.2) for tool-chain formalisms. The meta-models are extended to represent compositions of model flows, tool selections, and tool assessments based on software process engineering meta-models (SPEMs) [9]. In the previous work [32], the DSM models are transformed to visualization graphs which are used to support qualitative analysis of tool-chain performances. In the article, the DSM models are transformed to Bayesian network models in order to analyze the tool-chain performances using a quantitative way. When tool-chain developers build DSM models, several tool preference rules (introduced in Section 3.3) are defined to transform DSM models into Bayesian network models. Using a developed code generator in DSM tools, XDSL files are generated referring to tool assessment models based on Bayesian network. Then, tool assessment models are calculated through a Bayesian network model and used for quantitative tool assessment.

Table 2 Meta-meta models and meta-models supporting tool-chain formalisms

Meta-meta models	Meta-models	Description
Graph	Model flow	Representing model flow views of the tool-chain.
	Tool selection	Representing tool selection views of the tool-chain.
	Tool assessment	Representing tool assessment views of the tool-chain.
	Tool-chain structure	Representing tool-chain structures.
Object	Role (Tool-chain structure Graph)	A user of the tool-chain in the graph.
	Tool adapter task(Tool selection and Model flow Graphs)	A task of the tool adaptor, such as executing model transformations.
	Model (Tool selection and Model flow Graphs)	A model of the tool-chain in the graph.
	Tool(Tool-chain structure and Tool selection Graphs)	A tool of the tool-chain in the graph.
	Metrics (Tool assessment Graph)	A metric used for tool-chain measurement.
	Questions (Tool assessment Graph)	A question to define criteria for each metric.
Relationship	Control (Model flow and Tool selection Graphs)	The connections that track/control models.
	Data map (Model flow and Tool selection Graphs)	The connections that track/compare/synchronize models attributions.
	Co-simulation (Model flow and Tool selection Graphs)	The connections that execute co-simulations between models.
	Model transformation (Model flow and Tool selection Graphs)	The connections that track/compare/synchronize model information.
	Reference (Model flow and Tool selection Graphs)	The connections that track/compare/synchronize model versions.
	Refine(Tool selection Graph)	The connections between models and their tools.
	BNRelationship(Tool assessment Graph)	The connections between Bayesian network nodes used to connect metrics and questions.
	Control_channel (Tool-chain structure Graph)	The notification and invocation connections between tools and stakeholders.
	Data_channel (Tool-chain structure Graph)	The connections of model transformations between tools.
	Trace_channel (Tool-chain structure Graph)	The traceability connections between tools.
	Co-simulation_channel (Tool-chain structure Graph)	The co-simulation connections between tools.
Port	Input/Output ports of each relationship	The input and output points of related object relationships.
Role	From/To roles of each relationship	Defining connections between points and relationships.

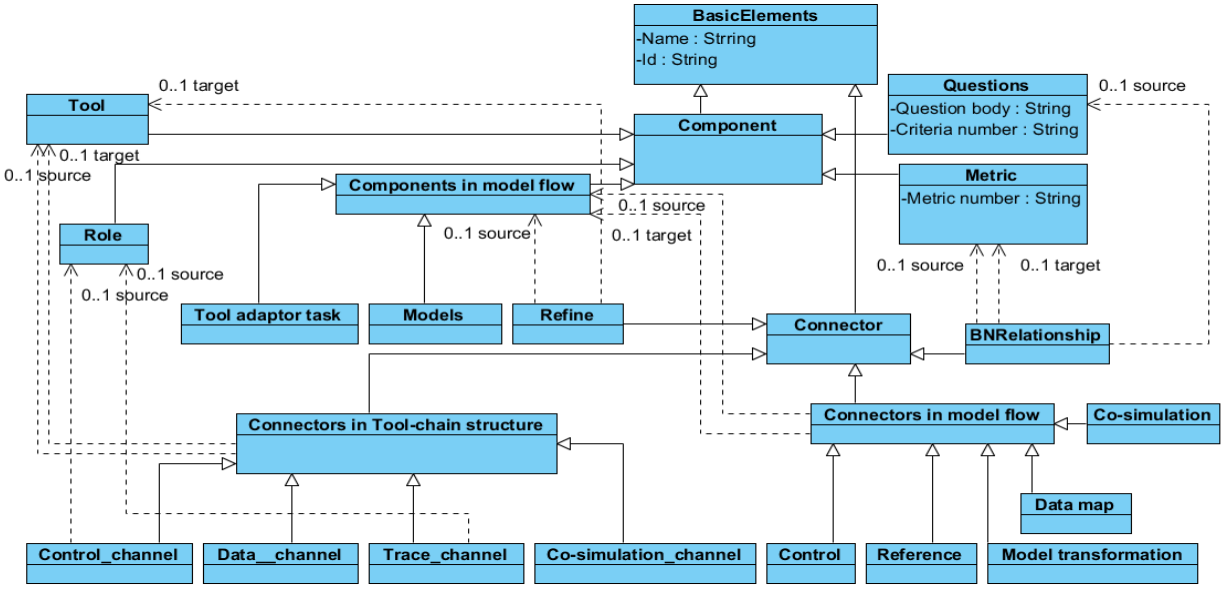

Fig. 2 Meta-models formalizing tool-chains

3.2. Domain-specific Modeling

3.2.1. GOPPRR Methodology formalizing Meta-models

In DSM tools, meta-models are defined to formalize model flows, tool selection rules, tool assessments and tool-chain structures based on GOPPRR meta-meta models [33]. They include **G**raph, (referring to one concept that UML class diagram integrated with the UML package, representing a collection of object and relationship), **O**bject, **P**ort, **P**roperty, **R**ole and **R**elationship. They allow for multiple representations to develop meta-models for tool-chain formalism as shown in Fig. 2. The details of each meta-model are introduced in Table 2.

Totally, there are four graphs to represent different views of MBSE tool-chain development. In each Graph, different Objects and Relationships are de fined to represent different views of tool-chains.

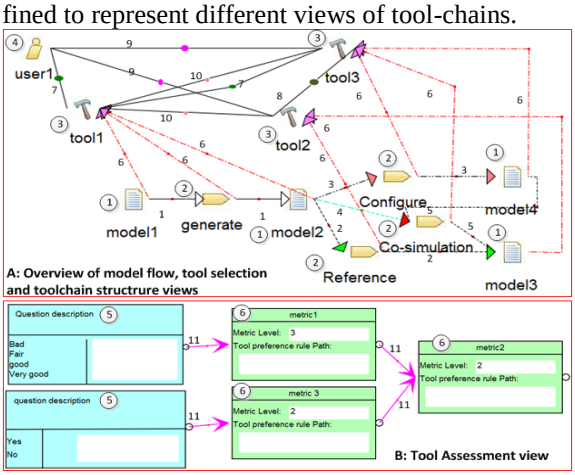

Fig. 3 A DSM model example illustrating the graphic syntax of language concepts. A:Overview of the model flow and tool-chain structure views; B: Tool selection view

The Objects and Relationships are defined based on compositions of tool-chains. For example, in the tool-chain structure graph (shown in Fig. 3 and Table 2), *Object Role* represents one user who makes use of one *Object Tool*.

3.2.2. Meta-models of tool-chain formalisms

The meta-models are extended based on a domain-specific modeling language for tool-chain descriptions (more details see previous research [10]) [8]:

- Four Graphs to represent an MBSE tool-chain: (1) model flow Graph to represent the model flows in a tool-chain; (2) tool selection Graph to define the tool used for design tasks and developing models; (3) tool assessments Graph to define questions and metrics for assessing tools; (4) tool-chain structure Graph to define tool compositions and their connections in the tool-chains.
- In each Graph, Objects, including *Role*, *Tool adapter task*, *Model*, *Metrics* and *Questions*, are separately used for representing different views of tool-chains whose definitions are shown in Table 2.
- Relationships are defined to represent interrelationships between Objects in a model flow Graph including (1) control; (2) reference; (3) model transformation; (4) data map; (5) co-simulation. They are used to describe connections between models and data in the tool-chains.
- Moreover, four channels are defined to describe connections between tools in the tool-chains. They are: (1) Control channel; (2) Data channel; (3) Trace channel; (4) Co-simulation channel; these channels refer to different interrelationships between tools.
- Two additional relationships: Refine and BNRelationship are used to represent the process of tool selections and assessment in the Tool selection Graph and Tool assessment Graph.

In Fig. 2, meta-models are described based on Table 2. The abstract syntax is introduced in order to show how different Objects connect with each other using Relationships. One example of the abstract syntax of related concepts is illustrated in Fig. 3. The circles with numbers represent concepts, such as ① represents models. The numbers represent the relationships between them.

The Fig. 3A presents overviews of the model flow, tool selection and tool-chain structure Graphs in three parts:

Model flow Graph: The Object representing components include 1) the *model* and 2) the *tool adapter task*. The connections include *1) model transformation*, *2) control*, *3) data map*, *4) reference* and *5) co-simulation*. In Fig. 3A, one example is shown how the model represents the model flow in a tool-chain (an example is shown in Section 0). The *model1* (①) is transformed to model2 through one tool adapter task (②) - generate (①). Then *model2* enables *model3 to* update its version through a task- *references* (using one adapter to implement model updates). Moreover, *Model2* updates attributes in *model4* as one *data map* through one *configure task*. Finally, *model2* controls co-simulations between *model3* and *model4* (one model in a master tool to implement *model3* during co-simulation).

Tool selection Graph: Compared with ***model flow Graph***, one additional component, *tools* (③) and an

additional connection *refine* (6) are added. *Refine* represents related models and tasks are implemented in a *tool*. In Fig. 3A, *model1*, the *tool adapter task-generate* and *model2* are implemented in *tool1*. The *Refine* connections between these elements and *tool1* represent they are implemented in *tool1*. Moreover, other *Refine* connections represent *model2* is implemented in *tool2* and the *tool adapter task Reference*, *model3*, the *configure task*, *model4* and relevant co-simulation are executed in *Tool3*.

Tool-chain structure Graph: The *tool-chain structure* ***Graph*** is constructed by *tool*③, *role* ④ and four tool-chain channels: 1) *control* (7); 2) *co-simulation* (8); 3) *trace* (9); and 4) *data channels* (10). This graph represents the relationships between tool-chains' compositions and relevant users. In Fig. 3A, one *user* controls *tool1* to trace *tool2* and *tool3*. It represents *tool1* updates models in *tool2* and *tool3* and controls *tool3* to implement co-simulations.

Fig. 3B shows one model based on the ***tool assessment Graph*** consisting of two components: *Questions* Object ⑤ and *Metric Object* ⑥; and one connector *BNRelationship* (*11)*. The model represents one assessment process based on Bayesian network theory [34]. The assessment process starts from basic *questions* for assessing each defined *metric*. In Fig. 3B, two types of questions are provided. Based on the probabilities of candidate answers in each question and defined preference rules, criteria in each metric are calculated. In different hierarchical levels of metrics, the *metric* Objects in low levels are used as inputs to the high-level *metric* Objects; e.g. *metric1* and *metric3* are inputs of *metric2* (in this case, *metric2* is at higher level than *metric3*).

3.3. Tool Assessment based on Bayesian Network

To assess tools using ***tool assessment view***, graphical probabilistic models based on Bayesian networks are used to represent random variables and their conditional dependencies of tool measurement. Through this approach, developers translate the complex relationships among metrics, questions and the measurement dependency criteria into visualized and mathematical models. Each node of the Bayesian network represents an individual metric or question with criteria. Links between nodes illustrate dependencies. In each metric, developers can configure their own preference rules as a **C**onditional **P**robability **T**ables (CPTs). The models support a multi-dimensional probability distribution for developers to analyze tool-chains quantitatively.

3.3.1. Bayesian networks

According to Bayes' rule,

$$P(A,B) = P(A \mid B)P(B) \tag{1}$$

The joint probability distribution over nodes can be factorized and represented by a Bayesian network. If there are *n* nodes $N_1, N_2, ..., N_n$, then

$$P(N_1, N_2, ..., N_n) = P(N_n \mid N_{n-1}, ..., N_1) \bullet P(N_{n-1} \mid N_{n-2}, ..., N_1) ... P(N_2 \mid N_1) \bullet P(N_1) \tag{2}$$

The factors in the factorized form are conditional probability distribution matrices. In our case, network nodes have two forms: questions and metrics. Questions are located only at the margin of the network.

Definition 1. A measurement refers to one process measuring tool-chain metrics. There are two types of measurement: (1) measuring metrics based on questions and (2) measuring metrics based on other metrics. A metric refers to one basic standard concept of tool-chain measurements, such as front-end usability. A criterion refers to one standardized scale in each metric and question. Each question criterion refers to one answer candidate and has a related criteria condition that refers to the conditions satisfying the scale. Each criterion in metrics refers to the related scale.

Definition 2. $Q_i^{j_i}$ refers to the i^{th} question ($i = 1, ..., I$) with j_i types of criteria regarding to the i^{th} question. $P(Q_i^{j_i})$ denotes the probability of the j_i^{th} k^{th}criterion of the i^{th} i^{th} question.

Definition 3. $M_a^{b_a}$ refers to the a^{th} metric in one measurement ($1 \le a \le A$) with the b_a criteria regarding the a^{th} metric. $P(M_a^{b_a})$ denotes the probability of the b_a^{th} criterion of the a^{th} metric.

When developers implement a measurement, in each question, they must input the probability distributions of each question criterion:

$$\begin{bmatrix} P(Q_1^1) & P(Q_1^2) & ... & P(Q_1^{j_1})... \\ P(Q_2^1) & P(Q_2^2) & ... & P(Q_2^{j_2})... \\ ... & ... & & ... \\ P(Q_I^1) & P(Q_I^2) & ... & P(Q_I^{j_I})... \end{bmatrix} \tag{3}$$

Detailed questions are introduced in Section 4.3.2.

Assuming that different questions are **I**dentically **I**ndependently **D**istributed (IDD), the probabilities of different questions are irrelevant to each other. Hence

the joint probability distribution of different questions can be simplified:

$$P(Q_1, Q_2, ..., Q_I) = P(Q_I \mid Q_{I-1}, Q_{I-2}, ..., Q_1) \bullet P(Q_{I-1} \mid Q_{I-2}, Q_{I-3}, ..., Q_1) ... P(Q_1) = \prod_{i=1}^{I} P(Q_i) \quad (4)$$

Substituting the denotations of questions and metrics into the overall formalization of the network using Eq. (2) and Eq. (4), the factorization can be simplified as

$$P(M_A, ..., M_1, Q_I, ..., Q_1) = P(M_A \mid M_{A-1}, ..., M_1, Q_I, ..., Q_1) \bullet P(M_{A-1} \mid M_{A-2}, ..., M_1, Q_I, ..., Q_1) ... P(M_1 \mid Q_I, ..., Q_1) \bullet \prod_{i=1}^{I} P(Q_i) \quad (5)$$

The probability distribution of questions can be defined as

$$\left[P(Q_i^1), P(Q_i^2), ... P(Q_i^{j_i}) \right] (1 \le i \le I) \quad (6)$$

and the conditional probability distribution of the a^{th} metric is also predefined as a CPT referring to a preference rule introduced in Section 4.3.3. The probability distribution of metrics can be defined as

$$P(M_A \mid M_{A-1}, ..., M_1, Q_I, ..., Q_1) \quad (7)$$

Nodes by the margin of the network (questions, in this case) have no predecessors and are characterized by their prior marginal probability distribution. Thus, any probability in the joint probability distribution is determined and calculated from these explicitly represented prior and conditional probabilities as shown in Eq. (5). According to the total probability formula, the probability of each criterion is presented as

$$P(M_a^{b_a}) = \sum_{z=1}^{b_{a-1}} P(M_{a-1}^{z}) P(M_a^{b_a} \mid M_{a-1}^{z}) \quad (8)$$

where z refers to the z^{th} criterion for the $(a-1)^{th}$ metric. The posterior probability of each criterion at each metric can be calculated in each node.

3.3.2. Metric Design

Using meta-models developed in Section 3.2, developers design a measurement using *Metric Object* and *Questions Object* in the *Tool Assessment Graph* in order to realize tool assessment. During developing a measurement, *Question Objects* are used to design different questions for tool-chain developers to answer. Based on the abstract syntax (how to connect different objects) defined in the *Tool Assessment Graph*, *Question Objects* in the margin are connected to different *Metric Objects.* The abstract syntax is defined, as shown in Fig. 3B, based on Bayesian network introduced in Section 3.3.1. Moreover, the topology between the final Metric Object (TMCL) and other Metric Objects is also developed based on the abstract syntax and Bayesian network.

The topology of metrics is derived from two main aspects: 1) Tool interoperability; 2) MBSE capabilities. The metrics are developed based on LISI [12] and metrics for MBSE tool trade-off [3]. They are designed from four main aspects to describe the abilities of a tool to support MBSE.

- ***Interoperability*** refers to: 1) the ability of two or more components in tool-chains to exchange data and models; 2) and the ability of tools to use the exchanged information in a heterogeneous network. When considering the interoperability, three questions are proposed for subjective assessment (details see Table 5).
- ***Social Procedures*** refers to the ability of tools to support management, security and usage of standards which are three views to assess the social procedure. Four questions are provided to support assessment of these views.
- ***Infrastructure*** refers to the capability that tools support tool-chain infrastructure. Front-end usability and system services are two main aspects which are needed to assess. Eight related questions are proposed to support these measurements.
- ***Application*** refers to the capability that tools support MBSE. This measurement is defined mainly based on MBSE tool trade-off [3]. Five views are considered and assessed based on eight questions.

Related metrics and questions are defined within these four aspects in Table 5. The questions are designed to calculate the probabilities for the criteria in the metrics. There are five scales which are defined for each criterion. After calculations, the scale with the highest probability is considered as the final criterion of the metric.

Table 3 Preference rule design

Does the tool support security policy in the company?		Yes				No			
Does the tool provide APIs for single sign-on and authentication?		Bad	Fair	Good	Very good	Bad	Fair	Good	Very good
Criteria in metrics	L1	0.7	0.3	0	0	1	1	1	1
	L2	0.3	0.7	0	0	0	0	0	0
	L3	0	0	0.4	0	0	0	0	0
	L4	0	0	0.6	0.1	0	0	0	0
	L5	0	0	0	0.9	0	0	0	0

3.3.3. Preference Rule Design

To implement the tool assessment, developers input tool preference rules as the CPTs used in the Bayesian network models. For example, two independent questions with two and four criteria are used to assess the security policy metric in Table 5. The preference rule is designed as shown in Table 3. In this situation, the question 'Does the tool support security policy in the company?' is more important to the metric than the other one. Therefore, if a negative answer is given, the metric is assessed as L1. Based on the preference rule design, different metrics are assessed based on the developers' preferences.

4. Case study

In order to verify our approach, a case study is provided including two MBSE tool-chains for auto-braking system design with automated verification using simulation. The problem statement is first introduced to present the user case of these tool-chains. Then the two tool-chains are presented in details.

4.1. Problem Statement

The purpose of the tool-chains is to test whether a new developed controller for an auto-braking system which can satisfy the "three-second rule", that is a measurement of the time interval at which vehicles should pass the same fixed point on the road. If a vehicle reaches such a point within three seconds after the vehicle in front of it, then the vehicle's following distance is too short. Aiming to test this problem, simulations for this situation are used to verify the performance of the auto-braking system which includes multi-domains, such as hydraulic system, control system, etc. In order to formalize the development process and system artifacts of the auto-braking system, DSM models are built which are also used to support automated simulation configurations.

From previous work, two tool-chain frameworks are developed: Social Process Information and Technique (SPIT) and Social Process Information seRvice Infrastructure and Technique (SPIRIT) [32]. These two frameworks support to construct these two tool-chains in different techniques: In the SPIT framework, tool-integration is realized using model transformations (introduced in Section 4.2). The SPIRIT tool-chain is developed based on the SPIRIT framework which tools are integrated using a service-oriented approach (introduced in Section 4.3).

Using our approach, these tool-chains are compared. Through the comparisons, the metrics measuring capabilities of the DSM approach are designed from two aspects: 1) Tool-chain formalism; 2) Tool assessment. The metrics and related criteria are introduced as follow:

- ***Tool-chain formalism:*** it includes four types of metrics: 1) *model flow view* with a criterion "Can the approach formalize model flows?"; 2) *tool selection view* with a criterion "Can the approach formalize tool selection?"; 3) *tool assessment view* with a criterion "Can the approach formalize tool assessment?"; 4) *tool-chain structure* view with a criterion "Can the approach formalize tool-chain construction?".
- ***Tool assessment:*** it includes two types of metrics: 1) *tool measurement design* with a criterion "Can the measurement be designed using DSM models?"; 2) *Tool measurement using a quantitative way* with a criterion "if the measurement can be evaluated in a quantitative way automatically?".

In order to formalize these two tool-chains as examples, the DSM approach is implemented to model tool-chain concepts first using MetaEdit+. It is adopted as a DSM tool to develop meta-models and build DSM models of the tool-chains [32]. Moreover, the DSM models are transformed to Bayesian network models which are used to calculate the metrics of tool-chains using a quantitative approach in GeNie. GeNie is a tool supporting Bayesian network calculations [35].

Table 4 Tools used in two tool-chains

Tool-chains	Tools	Users	Tool description	Purposes
Tools in Tool-chain A	Matlab/Simulink	Control engineer	A modeling tool support hybrid system simulation [38].	Develop a control system model and an integrated system model for co-simulation.
	MWorks	Multi-domain engineering, such as hydraulics	A modeling tool based on Modelica [39].	Develop multi-domain models
	Carmaker	Domain engineering for vehicle dynamics	A virtual simulation environment for vehicle dynamics (http://www.scribd.com/doc/17842366/IPG-CarMaker-Programmers-Guide#scribd).	Develop a vehicle dynamics model
	MetaEdit+	System engineer	A DSM tool based on GOPPRR [32].	Develop DSM models for an auto-braking system to formalize requirements, architecture and V&V
Additional tools in Tool-chain B	Service management tool	Project manager/IT engineer	A tool for transforming heterogeneous data to OSLC services[32].	Generate OSLC services for tool integration
	BPM Camunda –based web-based process management system (WPMS) generator	Project manager	A tool aims to transform DSM models to a web-based process management system [32].	Generate WPMS for process monitoring and control.

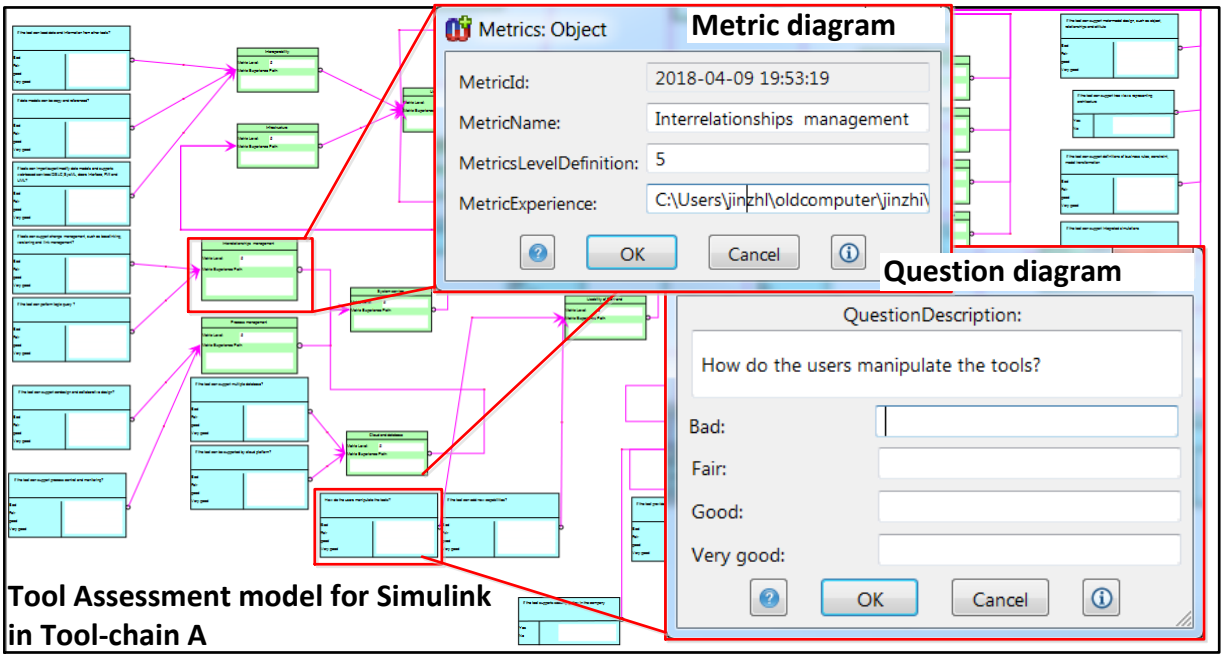

Fig. 4 Front-end for tool-chain developers to answer the questions

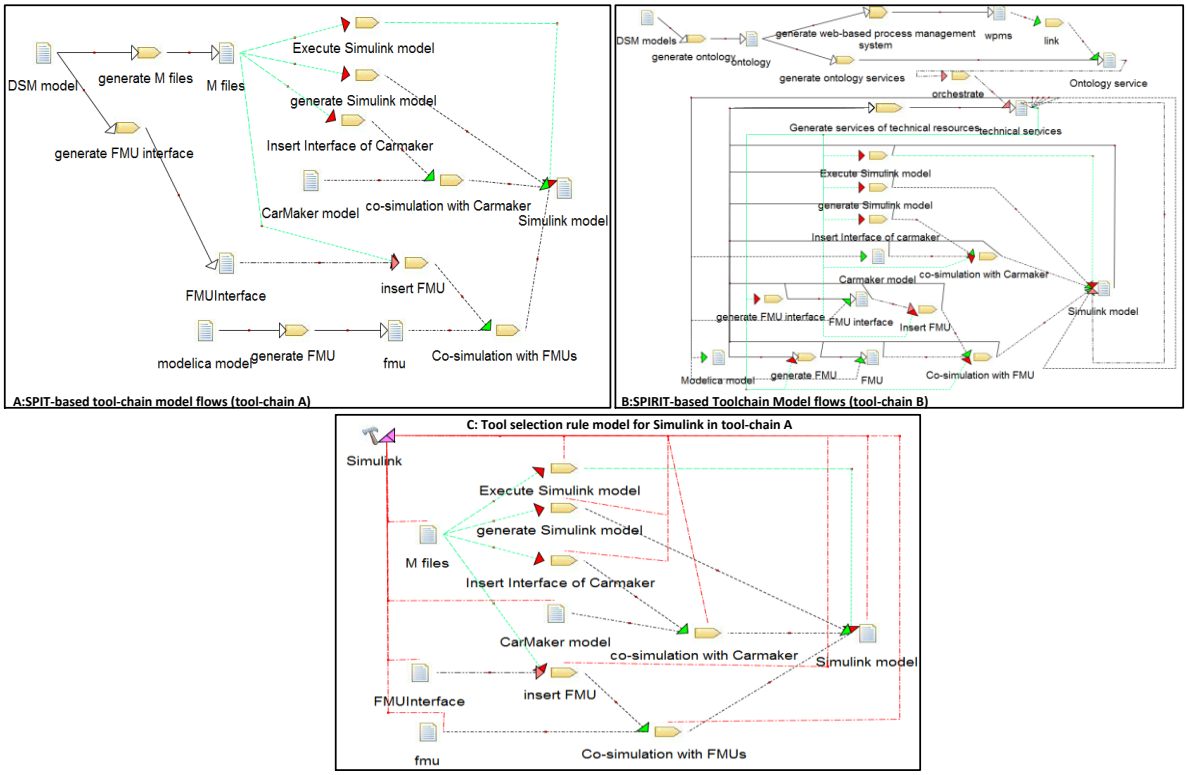

Fig. 5 A: Model flows of Tool-chain A; B: Model flows of *Tool-chain B*; C: Tool selection rule model for Simulink in Tool-chain A.

In MetaEdit+, meta-models are first developed based on Section 3.2.2. DSM models are built using such meta-models, in order to formalize the four views of the two tool-chains. Then, preference rules are designed based on empirical data of tool-chain developers. All the tools in these two tool-chains are assessed based on the same preference rules which are the basics for Bayesian network model calculation. Then the DSM models are transformed to XSDL files representing tool assessments using a code generator developed in MetaEdit+. Integrated with preference rules, the XSDL files are loaded and calculated based on the Bayesian network models by GeNie. Through the results, tool-chain developers analyze the MBSE tool-chains' performance and make decisions for selecting a tool-chain solution.

The tools used in these two tool-chains are shown in Table 4. The DSM models formalizing these two tool-chains are shown in Fig. 5. The details of each tool-chain are introduced in Section 0 and Section 4.3.

4.2. Tool-chain A based on the SPIT framework

In order to support the controller design for the auto-braking system development, *Tool-chain A* is proposed to formalize development process and system artifacts of auto-braking system and to support automated V&V [36]. To verify that the control logic can realize auto-braking when two vehicles are driving, co-simulation is adopted to integrate models from different tools and to execute integrated simulation using the system level simulation model. In Fig. 5A, the model flow of the *Tool-chain A* is presented. The DSM models are transformed to Matlab files and one configuration file for developing the interfaces of functional mocked-up units (FMUs) which are black boxes with a defined formats for being accessed by other tools [37] (the FMUs for co-simulation are developed based on standard functional mocked-up interface-FMI). Then the Matlab files [38] are used to create and configure a co-simulation model in Simulink. Carmaker models and FMUs are configured as blocks in Simulink manually. The details of tools used in *Tool-chain A*, such as MWorks (a Modelica

modeling tool) [39] and Carmaker, are introduced in Table 4.

The whole process is represented as a model flow as shown in Fig. 5A. When using the tool-chain to realize automated verification, system developers build DSM models to represent system information about auto-braking systems, such as requirements, architecture and configurations in the V&V in MetaEdit+. Then M files are generated from the DSM models which are control Simulink to create models and to execute simulations automatically.

Fig. 5C and Fig. 4 represent tool selection and tool assessment models separately for Matlab/Simulink in the Tool-chain A. In Fig. 5C, the red lines refer to connectors to represent Simulink is used for implementing the related tool operation tasks and models. In Fig. 4, the tool assessment of Simulink is formalized as a model based on Bayesian networks. Tool-chain developers answer the questions in the front-end of each question Object, define and configure the preference rule for each metric. When they finish the configurations of DSM models, they transformed the DSM models to Bayesian network models for quantitative analysis.

4.3. Tool-chain B based the SPIRIT framework

Another solution is *Tool-chain B* which is proposed to support co-simulations of auto-braking system based on the SPIRIT framework. The *Tool-chain B* is constructed by using a service-oriented approach aiming for promoting process management, tool-integration and interrelationships between different tools.

As shown in Fig. 5B, compared with *Tool-chain A*, two additional tools are used: a WPMS generator based on BPM Camunda (short for BPM Camunda) and a service management tool (short for service compiler). The service management tool aims to transform all the technical resources, such as models, data and tool APIs, to RESTful services [40]. The WPMS generator is used to transform the DSM models to a web-based process management system for developers to implement co-simulation configuration and execution automatically.

When stakeholders make use of the *Tool-chain B*, the model flow is shown in Fig. 5B. DSM models in MetaEdit+ are firstly used to generate ontology which refers to XML files representing information of development process and related system artifacts, such as requirement, system architecture. Then, the WPMS generator is used to load ontologies and generate related WPMS linked with RESTful services generated from technical resources. Developers manipulate and access technical resources through the WPMS using such services. The descriptions of these tools are demonstrated in Table 4.

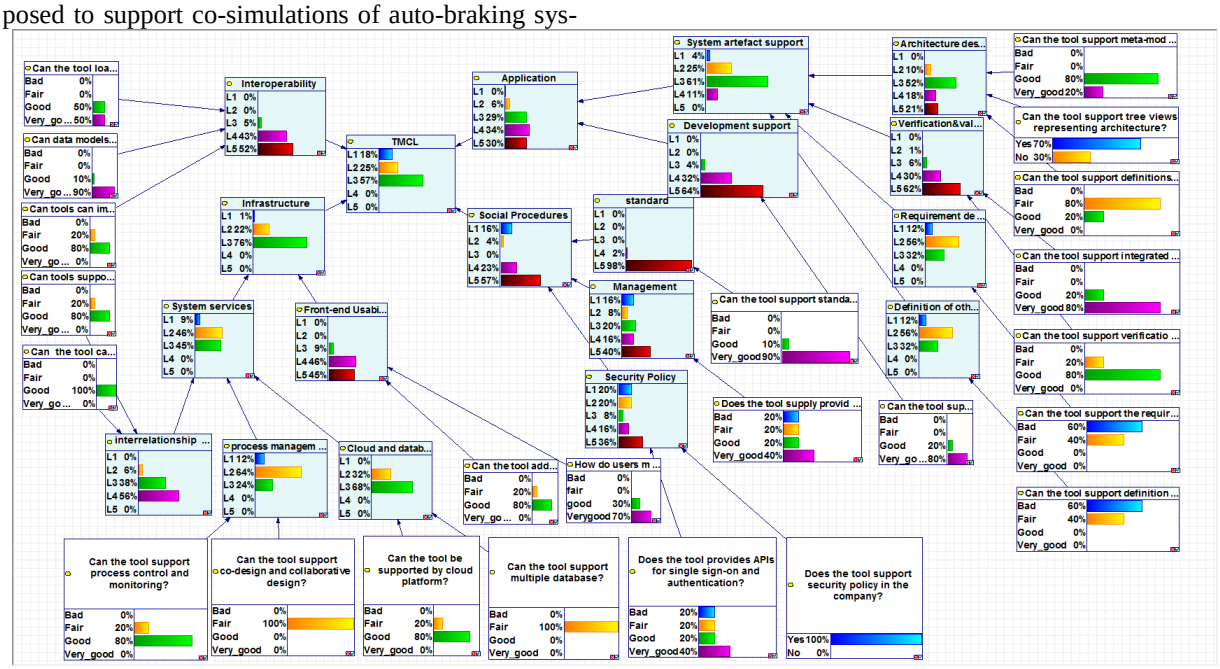

Fig. 6 Bayesian network models in GeNie (Black blocks refer to questions and green ones refer to metrics)

5. Results and Discussion

In order to evaluate our proposed approach, the process for comparing *Tool-chain A* and *Tool-chain B* is used to illustrate how the tool-chains are formalized and assessed. The objective measurements are implemented by the previous work using visualization [32]. Bayesian network models supporting quantitative analysis are used to implement the subjective measurements.

5.1. Quantitative Analysis

Using Bayesian network models, tools are assessed in GeNie models through calculating the metrics as shown in Fig. 6. The white nodes in the margin of the GeNie model represent the questions for stakeholders to answer. The parameters are configured based on the DSM models. After calculation, the probabilities of criterion value are represented in each node of the GeNie model. When tool-chain developers make use of the GeNie models, they assess the tool-chains through the criteria.

The margins of the models are the question nodes. They are calculated based on the questions nodes which developers input and defined preference rules. In each metric, five levels are defined as criteria. The probability distribution of each criterion is calculated in GeNie to provide developers a quantitative way to assess each metric. For example, in Fig. 6, the TMCL node represents the L3 is 57% which is larger than other levels. This means the TMCL is considered as L3.

As shown in Fig. 7, metrics of each tool in these tool-chains are demonstrated. Compared with *Tool-chain A*, two additional tools are adopted in *Tool-chain B*: 1) a service management tool (short for service compiler); 2) and a WPMS based BPM Camunda (short for BPM Camunda). Fig. 7 illustrates the comparisons between *Tool-chain A* and *Tool-chain B*. In *Tool-chain A-A*, the criteria of Matlab, MWorks, CarMaker and MetaEdit+ are shown. One example of TMCL criterion values of Matlab is illustrated in *Tool-chain A-B*. In *Tool-chain B-C*, the TMCL criteria of all the tools including additional two tools are shown. Moreover, the criterion values of the two tools are demonstrated in *Tool-chain B-D*.

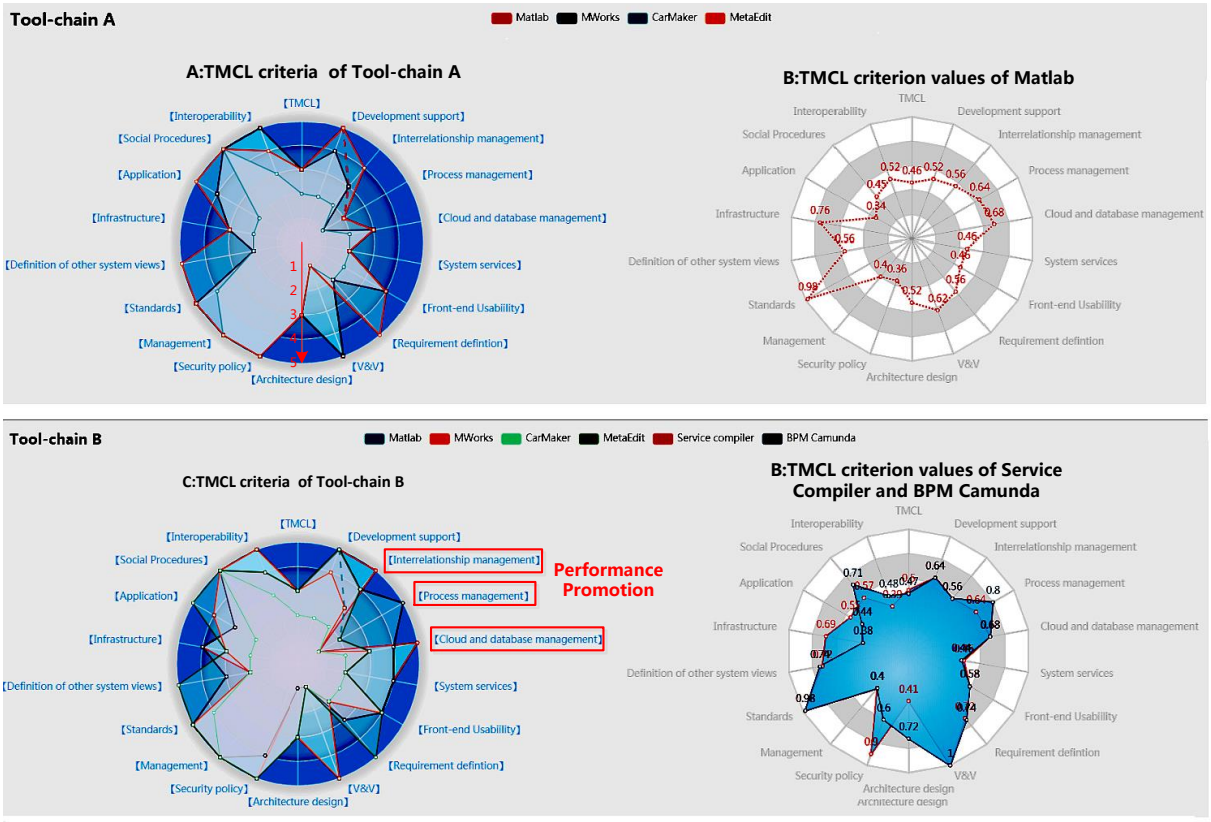

Fig. 7 Metrics and their final levels

5.2. Analysis from DSM models

From Fig. 7, we find the process management capability of BPM Camunda (red high lightened) has the L5 among the tools, because the generated web-based process management system supports process control and monitoring. The service compiler has the best interrelationship management and cloud database management (red high lightened), because the service compiler is used to generate the Restful services that can be accessed by other tools. Through results of Bayesian network models, some analysis results are provided. Then compared with the real empirical findings from pratices, the analysis results are verified.

After prototyping the *Tool-chain A* for co-simulation in this case study, some empirical findings were obtained. Although the *Tool-chain A* could support co-simulations and satisfy specific purposes, several limitations remain:

Process control and management: Though system developers can use DSM models to formalize, design, and describe co-simulation processes, such processes are difficult to control and manage as their complexity increases. For example, in some co-design scenarios, tool-chains are limited to managing technical resources across stakeholders during co-simulations.

Tool-integration: *Tool-chain A* can partially realize control and data integration. For example, the DSM model can generate Matlab language to control simulations in Simulink. However, the lack of supporting open standards makes this end-to-end integration inefficient. Integrations across design tools are difficult to realize.

Management of "interrelationships": With the help of DSM tools, interrelationships, i.e., the dependency and traceability of elements in DSM models can be managed. However, interrelationships are difficult to manage among technical resources. For example, relationships between DSM models and technical resources can be traced; however, traceability among technical resources is difficult to manage with additional tools' support, because of heterogeneous and unified representations of data, model and tool APIs.

Compared with these limitations, the *Tool-chain B* is developed based on the SPIRIT framework. The service compiler and BPM Camunda indeed improved related features of the updated tool-chain. From compared with the analysis from DSM models and real practices of these two prototypes, the DSM approach is verified that it can support:

- Tool-chain formalism: The DSM models can formalize 1) the model flow using model flow Graph; 2) tool-selection views using the tool-selection Graph; 3) tool assessment view using tool assessment Graph; 4) tool-chain structure using the tool-chain structure Graph.
- Tool assessment: The measurements of tool-chains can be designed based on metric and question Objects when developing the DSM model for tool-chains. Moreover, the generated Bayesian network models provide the quantitative results for tool measurement, as shown in Fig. 6.

5.3. Approach Extension to Large-Scale IT systems

Large-scale system is a term used in fields including computer science, software engineering and systems engineering to refer to software intensive systems with unprecedented amounts of hardware, lines of source code, numbers of users, and volumes of data [41]. In this paper, large-scale systems are defined as complex systems, such as aircraft and complex IT systems which are used to support complex system development. The approach is mainly used to support complex IT systems (MBSE tool-chains) for complex systems. In the case study, the MBSE tool-chain is proposed for 4 types of roles (Fig 6. in [32]). One role is domain engineers referring to the stakeholders from different domains of auto-braking system development, such as mechanical domains. During the whole lifecycle of the auto-braking systems, stakeholders construct one large social network. Moreover, using the given tools, they develop different models for different views. In this situation, auto-braking system is considered as one large-scale system whose development processes, including different models, stakeholders and system views, are also large scale systems.

During designing large-scale systems, Hillary G Sillitto proposed 4 key points to guide design activities [42]:

1). "Finding pragmatic and relevant measures of stability margin". In our approach, the measurement is designed based on LISI [12] and metrics for MBSE tool trade-off [3].

LISI is a standard to assess the maturity of IT systems for complex system [12]. The metrics in LISI are designed from four aspects: 1) procedures; 2) data; 3) application; 4) infrastructure where IT systems are considered as social-technical systems. The metrics in this paper are designed based on LISI and

MBSE trade-off rules ([3]) in Boeings supporting assessment of MBSE tool-chains from not only tool-chain performances, but also policies and cultures.

2). "Policies, values, behaviors and cultures, and emergence are required to consider". MBSE tool-chains and their stakeholders construct a socio-technical system. During designing the metrics of this system, four aspects are considered, including interoperability , social procedures , infrastructure and application.

3). "Select a modeling approach and framework to identify system challenges". We make use of a GOPPRR approach and Bayesian networks to support formalize tool-chains and assess tool-chains using a quantitative approach.

Currently, model-based design is one important solution to support large-scale system development. Boeings has already made use of model-based systems engineering to support architecture design of integrated system in aircrafts [43]. From their insights, models are key important basis to describe the architecture of large-scale systems, not only complex systems, but also complex IT systems [44]. Since the model maturity is increasing, the completeness, correctness and accuracy of architectures are growing. Then when the systems are developed, developers can capture enough information before system development.

4). "Model driven architecture provides a closed system boundary for the large-scale systems". We make use of code generation to support automated assessment of MBSE tool-chains.

Model-driven technique is also proposed as one solution to support large-scale system development and large-scale projects [45]. These techniques can realize automated verification from architecture models. In this way, the efficiency of verifying large-scale systems is promoted.

In summary, this paper proposed a GOPPRR approach to support MBSE tool-chain formalisms and assessment. The GOPPRR approach is a powerful method for modeling architectures [46]. Using GOPPRR approach, meta-models for formalizing MBSE tool-chains are developed based on previous work [11]. These meta-models can also be extended in order to satisfy the demand of large scale systems. Using code generation, the DSM models formalizing MBSE tool-chains are transformed to Bayesian network models for quantitative analysis automatically. Finally, each tool's TMCL is identified. This process can promote the efficiency of large scale MBSE tool-chain assessment.

5.4. Summary and Limitations

From the case study, we find the *model flow view*, *tool selection view*, *tool assessment view*, *tool-chain structure view* are formalized using DSM models. Tool measurement is designed as meta-models based on TMCL. Evaluation of tool measurement is implemented by BN models. Our approach can satisfy the demands of the metrics mentioned in Section 4.1. The four views of tool-chain can be formalized using DSM models in MetaEdit+. Based on Bayesian network theory, DSM models formalize measurements of related tools and are transformed into Bayesian network models in GeNie. Then GeNie calculates related models and evaluates tool measurements. Developers compare different tool-chains and select tools using a quantitative approach. In our previous work, DSM models are also transformed to tool-chain visualizations [32]. Through visualizations, three non-subjective measurements of the tool-chains are analyzed. Tool-chain developers use such views to analyze the tool-chains' features, compare solutions, and verify concepts in early phases.

Using our approach, developers can visually design tool-chain solutions, leading to lowering research and development costs compared with a set of documents. Reusable preference rules can promote efficiency and confidence in tool assessment. Through comparisons, our approach can support developers in formalizing assorted views of their tool-chains using DSM models, to assess tool performances using Bayesian networks. However, our approach has several limitations:

- The TMCL measurement was designed relative to MBSE tools. The metric and criteria are defined to focus on MBSE techniques. Further measurement will be done to provide more comprehensive criteria for other tool-chain domains, such as product lifecycle or data management systems.
- Economic factors of tool-chain are not considered, such as the cost-efficiency of tools.

6. Conclusion and Future Work

In this paper, a DSM approach is proposed to support MBSE tool-chain development by assessing the MBSE tool-chain concepts before prototyping. The DSM approach formalizes tool-chain concepts, assesses tools via quantitative analysis based on Bayes-

ian networks in order to promote the confidences of tool-chain developers.

In conclusion, our approach makes two main contributions:

- The DSM approach formalizes four views of tool-chain: (1) model flow; (2) Tool selection; (3) Tool assessment; (4) Tool-chain structure. From these views, tool-chain developers promote their confidences and understanding in order decrease the risks led by failures of tool-chain development.
- Using Bayesian network theory introduced in Section 3.3.1, subjective measurements can be implemented and analyzed quantitatively. The previous work only supports objective measurements of MBSE tool-chains [32]. The results generated from Bayesian network models provide clues for comparing different tool-chains. Tool-chain developers make use of such measurements to analyze tool-chain concepts and make decisions. Compared with state-of-the-art, Bayesian network models are designed to support MBSE tool-chain assessment rather than complex systems and IT systems. Moreover, after formalizing MBSE tool-chains, DSM models are directly transformed to Bayesian network models in order to realize the early verification automatically.

Given these results, our approach indeed promotes tool-chain development efficiency and provides a quantitative method for analyzing tool-chain features. In the current enterprises, MBSE tool-chains are expected to be large-scale IT systems support complete the system development in the future [1]. When developing tool-chains, early evaluation is very important before prototyping which allows tool-chain developers verify their products in order to avoid R&D cost led by the failures of tool-chain development.

In the future, standards will be used to support metrics design. Further research will focus on formalisms of tool-chain evolutions to describe its dynamics. A learning curve of each metrics (designed based on Markov chain and Monte Carlo) will be considered as one factor that impacts on effects of tool-chains.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments on this paper. The authors acknowledge financial support from the National Ministries (JCKY2014602B007).

References

[1] International Council on Systems Engineering, A World in Motion: Systems Engineering Vision 2025, *International Council on Systems Engineering*, (2014).

[2] Lu, Jinzhi , Chen, Dejiu , Gürdür, Didem. and Törngren, Martin (2017), An Investigation of Functionalities of Future Tool‐chain for Aerospace Industry, in: *Proceedings of the 27th INCOSE International Symposium* **27**(1)(2017), 1408-1422.

[3] Brittany Friedland, John Herrold, Glendora Ferguson, Robert Malone, Conducting a Model Based Systems Engineering Tool Trade Study Using a Systems Engineering Approach, in: *Proceedings of the 27th INCOSE International Symposium* **27**(1) (2017), 1087-1099.

[4] Hongming Cai, Mengwei Shi, Boyi Xu, and Mingjiu Yu, Semantic annotation for web3D scene based on three-layer ontology, *Integrated Computer-Aided Engineering* **22**(1)(2015), 87-101.

[5] Weiming Shen and Douglas H. Norrie, Dynamic manufacturing scheduling using both functional and resource related agents, *Integrated Computer-Aided Engineering*, **8**(1)(2001), 17-30.

[6] Lin H K and Harding J A, A manufacturing system engineering ontology model on the semantic web for interenterprise collaboration, *Computers in Industry* **58**(5) (2007), 428-437.

[7] Kootbally, Z., Schlenoff, C., Antonishek, B., Proctor, F., Kramer, T., Harrison, W., Downs, A., and Gupta, S.K., "Enabling Robot Agility in Manufacturing Kitting Applications," *Integrated Computer-Aided Engineering*, 25:2, 2018, pp. 193-212.

[8] Craig Schlenoff, Stephen Balakirsky and Henrik Christensen, An integrated semantic framework for designing context-aware Internet of Robotic Things systems, *Integrated Computer-Aided Engineering*, 2017 (Preprint): 1-20.

[9] Biehl M, El-Khoury J, Loiret F, Martin Törngren, On the modeling and generation of service-oriented tool chains, *Software & Systems Modeling* **13**(2)(2014), 461-480.

[10] H. Sun, W. Fan, W. Shen, T. Xiao and Y. Chai, Ontology maintenance in high level archi-

tecture federation development and execution process, *Integrated Computer-Aided Engineering* **20**(1) (2013), 79–94.
[11] Lu J, Törngren M, Chen D J, J. Wang, A Tool Integration Language to Formalize Co-simulation Tool-Chains for Cyber-Physical System (CPS), in: *Proceeding of 2017 International Conference on Software Engineering and Formal Methods* (2017) 391-405.
[12] CMMI Product Team, CMMI for Development, Improving processes for developing better products and services (2006).
[13] US DoD, Levels of Information Systems Interoperability (LISI), US DoD, (1998).
[14] Mankins J C. Technology Readiness Levels, (1995).
[15] Dassisti, M., Panetto, H., Tursi, A., and De Nicolo, M., Ontology-based model for production-control systems interoperability, in: Proceding of 5th CIRP Digital Enterprise Technology Conference (2008), 527-543.
[16] Osterloh, Andrea, Tool Chain Analysis Method, First Tool Qualification Symposium, (2013), Available from: http://www.validas.de/TQS/2013/abstracts/Slides_Osterloh.pdf
[17] Bustillo A, Grzenda M, Macukow B, Interpreting tree-based prediction models and their data in machining processes, *Integrated Computer-Aided Engineering* 23(4)(2016), 349-367.
[18] L. Pan, G. Paun, G. Zhang, F. Neri, "Spiking Neural P Systems with Communication on Request", *International Journal of Neural Systems*, vol. **27**, no. 8, 1750042, 2017
[19] T. Wu, F. Bible, A. Paun, L. Pan, F. Neri, "Simplified and yet Turing universal spiking neural P systems with communication on request", *International Journal of Neural Systems*, vol. **28**, no. 8, 1850013, 2018
[20] Huang, Y., Beck, J.L., and Li, H. (2019), "Multitask sparse Bayesian learning with applications in structural health monitoring," *Computer-Aided Civil and Infrastructure Engineering*, 34:9, 732-754.
[21] Yuen, K.V., Kuok, S.C., Dong, L. (2019, "Self-calibrating Bayesian real-time system identification," Computer-Aided Civil and Infrastructure Engineering, 34:9, 806-821.
[22] Liang, X. (2019), "Image-Based Post-Disaster Inspection of Reinforced Concrete Bridge Systems Using Deep Learning with Bayesian Optimization," *Computer-Aided Civil and Infrastructure Engineering*, 34:5, 415-430.
[23] Luo, X., Li, H., Yang, X., Yu, Y., and Cao, D. (2019), "Capturing and Understanding Workers' Activities in Far-Field Surveillance Videos with Deep Action Recognition and Bayesian Nonparametric Learning," *Computer-Aided Civil and Infrastructure Engineering*, 34:4, 333-351.
[24] Zhang, Y.,Wang, Y., Jin, J., and Wang, X. "Sparse Bayesian Learning for obtaining sparsity of EEG frequency bands based feature vectors in Motor Imagery Classification," *International Journal of Neural Systems*, 27:2, 2017, 1650032 (13 pages).
[25] Austin, M. F., Homberger, C., Ahalt, V., Doolittle, E., Polacek, G. A., and York, D. M. Applying Bayesian Networks to TRL Assessments–Innovation in Systems Engineering in: *Proceedings of the 27th INCOSE International Symposium* 27(1) (2017), 1622-1634.
[26] E. Nashnush and S. Vadera, "Learning Cost-Sensitive Bayesian Networks via Direct and In-Direct methods," Integrated Computer-Aided Engineering, 24:1, 2017, pp. 17-26.
[27] Schetinin, V., Jakaite, L., and Krzanowski, W., "Bayesian Learning of Models for Estimating Uncertainty in Alert Systems: Application to Air-craft Collision Avoidance," *Integrated Computer-Aided Engineering, 25:3, 2018, pp. 229-245.*
[28] L. Tan and S. Xu, L. Tan and S. Xu, "A model-checking-based approach to risk analysis in supply chain consolidations," *in Integr. Comput. Aided. Eng.*, vol. 16, no. 3, pp. 243–257, (2009).
[29] S. Pellegrinelli and N. Pedrocchi, "Estimation of robot execution time for close proximity human-robot collaboration," *in: Integr. Comput. Aided. Eng*., vol. 25, no. 1, pp. 81–96, (2017).
[30] R. E. Banchs, H. Klie, A. Rodriguez, S. G. Thomas, and M. F. Wheeler, "A neural stochastic multiscale optimization framework for sensor-based parameter estimation," *in: Integr. Comput. Aided. Eng.*, vol. 14, no. 3, pp. 213–223, (2007).
[31] S. Donnay, G. Gielen, and W. Sansen, "High-Level Power Minimization of Analog Sensor Interface Architectures," *in: Integr. Comput. Aided*. Eng., vol. 5, no. 4, pp. 303–314, (1998).
[32] Lu J, Gürdür D, Chen D J, et al. Empirical-Evolution of Frameworks Supporting Co-simulation Tool-Chain Development, in Proceeding of *2018 World Conference on Information Systems and Technologies* (2018): 813-828.
[33] Kelly S, Lyytinen K, Rossi M, Metaedit+ a fully configurable multi-user and multi-tool case and came environment, In: *Proceeding of Inter-*

national Conference on Advanced Information Systems Engineering (1996), 1-21.
[34] Friedman N, Geiger D, Goldszmidt M. Bayesian network classifiers, *Machine learning*, **29**(2-3) (1997), 131-163.
[35] Druzdzel M J, SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models, in: Proceeding of *16th National Conference on Artificial Intelligence* (1999), 902-903.
[36] Jinzhi L, Chen D, Törngren M, et al. A model-driven and tool-integration framework for whole vehicle co-simulation environments, in: *Proceeding of 8th European Congress on Embedded Real Time Software and Systems (ERTS)* (2016).
[37] MODELISAR Consortium, Functional Mock-up Interface for Co-Simulation (2010), 1-213.
[38] Natick, M A, The mathworks, available from: https://ww2.mathworks.cn/matlabcentral/.
[39] Chen X, Wei Z. A new modeling and simulation Platform-MWorks for electrical machine based on Modelica, in: *Proceeding of International Conference on Electrical Machines and Systems* (2008), 4065-4067.
[40] Z. Sun, J. Shen and J. Yong, A novel approach to data deduplication over the engineering-oriented cloud systems, *Integrated Computer-Aided Engineering* **20**(1) (2013), 45–57.
[41] L. Northrop, P. Feiler, R. P. Gabriel, J. Goodenough, and E. Al., "Ultra-large-scale systems - The Software Challenge of the Future," in: *companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, (2006).
[42] H. G. Sillitto, "Design principles for Ultra-Large-Scale (ULS) systems," in: "*20th Annual International Symposium of the International Council on Systems Engineering"*, (2010).
[43] R. Malone, B. Friedland, J. Herrold, and D. Fogarty, "Insights from Large Scale Model Based Systems Engineering at Boeing," in: "*INCOSE Int. Symp."*, vol. 26, no. 1, pp. 542–555,(2016).
[44] S. Izukura, K. Yanoo, T. Osaki, H. Sakaki, D. Kimura, and J. Xiang, "Applying a Model-Based Approach to IT Systems Development Using SysML Extension," in: *MODEL*, vol. 6981, pp. 563–577 (2011).
[45] V. Arnould, "Using model-driven approach for engineering the System Engineering System," in: *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, no. June, pp. 608–614 (2018).
[46] H. Kern, A. Hummel, and S. Kühne, "Towards a comparative analysis of meta-metamodels," in: *Proceedings of the compilation of the co-located workshops on DSM'11*, TMC'11, (2011).

Table 5 Metrics designed to support TMCL assessment

Metrics (Five Levels defined as criteria, L1-L5 which L5 is the highest Level.)				Questions	Criteria
TMCL	Interoperability			Can the tool load data and information from other tools?	Bad/Fair/Good/Very good
				Can data models be copied and referenced?	Bad/Fair/Good/Very good
				Can tools import/expert/modify data models and supports web-based services/OSLC, SysML, doors interface, FMI and UML?	Bad/Fair/Good/Very good
	Social Procedures	Standards		Can the tool support standards used in the domain?	Bad/Fair/Good/Very good
		Management		Does the tool supply provide enough training?	Bad/Fair/Good/Very good
		Security/policy		Does the tool support security policy in the company?	Yes/No
				Does the tool provides APIs for single sign-on and authentication?	Bad/Fair/Good/Very good
	Application	System artifact support	Architecture design	Can the tool support meta-model design, such as object, relationships and attitude?	Bad/Fair/Good/Very good
				Can the tool support tree views representing architecture?	Yes/No
				Can the tool support definitions of business rules, constraint, model transformation?	Bad/Fair/Good/Very good
			V&V	Can the tool support integrated simulations?	Bad/Fair/Good/Very good
				Can the tool support verifications of other views?	Bad/Fair/Good/Very good
			Requirement definition	Can the tool support the requirement definition?	Bad/Fair/Good/Very good
			Definition of other views	Can the tool support definition of other views?	Bad/Fair/Good/Very good
		Development support		Can the tool support the whole life cycle?	Bad/Fair/Good/Very good
	Infrastructure	Front-end Usability		How do users manipulate the tools?	Bad/Fair/Good/Very good
				Can the tool add new capabilities?	Bad/Fair/Good/Very good
		System services	Cloud and database manage-ment	Can the tool be supported by cloud platform?	Bad/Fair/Good/Very good
				Can the tool support multiple database?	Bad/Fair/Good/Very good
			process management	Can the tool support process control and monitoring?	Bad/Fair/Good/Very good
				Can the tool support co-design and collaborative design?	Bad/Fair/Good/Very good
			interrelationship management	Can the tool perform logic queries?	Bad/Fair/Good/Very good
				Can tools can support change management, such as base linking, versioning and linking management?	Bad/Fair/Good/Very good