This is the accepted version of a paper presented at *IEEE – 13th System of Systems Engineering Conference*.

Citation for the original published paper:

Towards A Service-oriented Framework for MBSE Tool-chain Development
In:

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-226673
Towards A Service-oriented Framework for MBSE Tool-chain Development

Jinzhi Lu*, DeJiu Chen*, Jian Wang†, Martin Torngren*
* KTH Royal Institute of Technology, 100 44, Stockholm
† University of Electronic Science and Technology, 611731, China

Email: jinzhl@kth.se

Abstract—This paper proposes a SPIRIT framework supporting model-based systems engineering (MBSE) tool-chain development of advanced cyber-physical systems (CPS) with emphasis on tool integration, process management, automated verification and validation. The core features of the developed MBSE tool-chain include domain-specific modeling to describe CPS development, service-oriented deployment of technical resources (data, model and tool operations) and process management through IT platforms. The framework has two purposes: to support tool-chain development with a systems engineering approach; to promote interoperability of the whole developed tool-chain through a service-oriented approach. The framework covers social, process, information and technical aspects aiming to integrate various related MBSE techniques with tool-chain development. Based on the framework, an MBSE tool-chain prototype is developed, and the flexibility and interoperability are evaluated through a case study.

Keywords: MBSE; Tool-chain development; Process management; Tool integration; Service orchestration

I. INTRODUCTION

Model-based Systems Engineering (MBSE) is proposed to deal with the increasing complexity of CPS. However, current MBSE tools are focused on specific domains, developed based on isolated and unrelated modeling theories and for specific purposes with different semantic models and uncommon storage representations - all of which challenge seamless integration between tools [1]. Moreover, heterogeneous and insufficient APIs provided by tools make platform, presentation and control integrations difficult across the whole life cycle [2]. Given the current status of MBSE, one of the challenges is integration across disciplines during the whole life cycle [3]. MBSE tool-chains are considered as a good solution to deal with this integration [4], and views of platform, presentation, data, control and process need to be considered during development [5].

In addition to the integration issues, MBSE tool-chain development is also challenged by domain-specific views, process management and automated verification & validation (V&V). Since CPS consists of multi-domain systems cross-domain software and hardware [6], focusing on different composite systems (their system characteristics), there is need to formalize them based on models using related domain-specific views. Moreover, the growing complexity means that development processes are no longer flexible, efficient and agile [7]. Particularly during a co-design and collaborative process, deployment of technical resources related to MBSE are much more complex than before. In addition, automated V&V, configuration and change management need to be integrated with process management in order to promote efficiency of CPS development.

In response to these challenges, we propose a service-oriented framework called SPIRIT to support MBSE tool-chain development. It consists of four layers(Social, Process, Information & seRvice Infrastructure (IRI) and Technical) and a learning-before-doing approach based on systems engineering. In addition, we adopt a service-oriented technique to integrate domain-specific modeling with process management, deployment of technical resources and automated verification and validation to for better interoperability.

In this paper, we present our solution in detail. We describe related work in Section II and the SPIRIT framework in Section III. Section IV introduces an MBSE tool-chain prototype developed based on the SPIRIT framework. We discuss the case study in Section V and offer our conclusions in Section VI.

II. RELATED WORK

A. Domain-specific modeling

Domain-specific modeling (DSM) is an engineering approach to describe facets of system abstractions through DSM models built based on domain-specified modeling language (DSML) to describe information for specific domains and systems through a set of formal meta-models [8], e.g., EAST-ADL [9], VHDL [10]. Developers make use of such meta-models to create DSM models to formalize related information about CPS products and their development.

B. Tool integration

Tool integration refers to activities that produce an integrated environment supporting the development process of CPS by sharing and reusing artifacts generated from different tools, tracing between such artifacts and accessing them from different stakeholders across the whole life cycle [11]. As summarized from existing solutions from the technical view, four types of tool integration are shown as follows:
• Linked data is used to describe a recommended best practice for exposing, sharing and connecting data, information, and knowledge on semantic webs using URIs and RDF, such as OSLC [12].
• Neutral data exchange is a standard-based approach for information exchange in favour of both the tool suppliers and the organizations that use multiple design tools within a whole life cycle, such as STEP AP233 [13].
• Meta-model integration refers to an approach facilitating model and data integration using transformations of meta-model and meta-data between tools such as [14].
• Tool-based integration refers to a specific tool providing related services enabling integration of models and APIs of tools such as ModelBus [15].

C. Process management based on IT platforms

Business process management (BPM) refers to a discipline to discover, model, analyze, improve and automate business processes using IT platforms [16]. Currently, there are several options to support business process modeling, e.g. activity diagram in UML [17], BPMN [18]. After the business process is formalized by such models, process engines can be implemented to generate related management systems for process control and monitoring, such as BPM Camunda [19].

In this paper, we concentrate on a framework for MBSE tool-chain development. First, the framework captures the tool-chain developers’ requirements for the target MBSE tool-chain, based on a systems engineering approach. Based on a SPIRIT (Social, Process, Information and Technical layers) framework [4] in which one MBSE tool-chain is considered as one system whose architecture description includes viewpoints capturing related stakeholders’ concerns. In contrast to SPIRIT, SPIRIT makes use of an IRI layer to replace the Information layer shown as Fig. 1. We summarize the concepts in each layer as follows:

The Social layer involves an explicit network of the different stakeholders (and the relationships between them) using the MBSE tool-chain. The stakeholders are related to the development processes implemented by the MBSE tool-chain, and address views in architecture descriptions of their target products. They also implement their own tasks and access the corresponding technical resources (models, data and tools). The implicit factors in the social layer are the rulers and policies of such social networks, the concerns of tool-chain developers and the environment of the related tool-chains.

In the Process layer, development processes related to the target products are considered. Each development process consists of Work Task concepts including Human Work Task and Automated Work Task, and Gate concepts representing decision making of the work flows and relationships between them.

In the IRI layer, views covered by the system architecture descriptions of related products are considered, such as requirement, model structure and function, etc. They represent the information of system artifacts in each Work Task (e.g. what’s the requirement?...) and deployment of related technical resources. Moreover, such information and technical resources are wrapped as services in a service infrastructure which can be accessed through URL (Introduced in Section C).

The Technical layer covers ontology, modeling methods and technical resources. Ontology refers to explicit formal standard instructions about development processes in the Process layer and information of system artifacts (system views) in the IRI layer.
layer. Modeling methods (how to model) are considered based on development process and related views, e.g. simulation in Matlab/Simulink is used for verification & validation (V&V) (introduced in Section IV). The technical resources include data, model, codes and tool APIs supporting development process and system views and enabling modeling methods.

B. Tool-chain Concepts Developed based on SPIRIT

After capturing the requirements of the MBSE tool-chain, a concept is proposed to integrate domain-specific views, tool-integration and process management based on a service infrastructure. In Fig.2, the workflow of the tool-chain concept is proposed:

1. In the social networks, stakeholders who work in the development process and have related views build DSM models using related tools. Therefore, the social network (red boxes) includes:
   - Stakeholders participating in the product development (Part 1.1).
   - DSM tools used to build models to represent the development processes and information (system views) of system artifacts (Part 1.2).
   - Web-based process management system (WPMS) embedded with services of information and technical resource deployment for stakeholders to implement their work tasks (Part 1.3).
2. Development processes are formalized by the process pattern (Part 2.1) of DSM models.
3. System views related to development processes are formalized by the information pattern (Part 2.2) of DSM models, such as model structures, V&V.
4. Through a developed code generator, DSM models are transformed to ontology in XML (Part 4.1) including process pattern (Part 4.1.1) and information pattern (Part 4.1.2).
5. Technical resources are used to support CPS development as follows:
   - Models (Part 4.1) supporting work tasks, e.g. Simulink models.
   - Data (Part 4.2) used for work tasks.
   - Tools (Part 4.3) used for work tasks, in particular, providing APIs.
6. A service compiler is used to generate a WPMS with related services of technical resources and system artifact information.
   - A process engine (Part 3.2.1) loads process pattern to generate a WPMS with related work tasks.
   - Service adapters (3.2.2) generate the corresponding services of information of system artifacts and deployment of the related technical resources.
   - A service orchestration (3.2.3) loads ontology to manage, configure and orchestrate services to the related providers for work tasks in the development process (introduced in Part D).
7. A service infrastructure (3.3) refers to a collection of services accessing related information and deployment of technical resources.
8. The service orchestration can generate service providers linked to the related work tasks in the WPMS. Moreover, such service providers also link the required services of information and technical resources in the service infrastructure.
9. Finally, stakeholders implement related development processes based on WPMS.

C. DSM Supporting MBSE tool-chain

In order to formalize the development processes and system views, stakeholders first design meta-models based on meta-meta models in DSM tools. For example, MetaEdit+ provides a GOPPR approach with formal meta-meta model concepts.
The DSM models include process patterns and information patterns. As shown in Fig.2, the process patterns represent CPS development processes and the information patterns describe information of system artifacts (system views) in each work task. We propose several definitions to support meta-model development. Fig.4 represents the packages of requirement, function, system architecture and V&V. In this paper, we adapt the GOPPR approach to develop the initial meta-models\(^1\).

**Definition 1:** A process pattern \(a\) refers to a Graph\(^2\), \(PM_a\):

\[
PM_a = \text{Set}(\text{Set}(\text{WorkTask}_{\text{Human}}_i, i \geq 0), \text{Set}(\text{WorkTask}_{\text{Auto}}_n, n \geq 0), \text{Set}(\text{Seq}_m, m \geq 0), (\text{Set}(\text{Gate}_k, k \geq 0), \text{start}, \text{end}))
\]

where \(\text{Set}()\) refers to one collection; The Object\(^3\) \(\text{WorkTask}_{\text{Human}}_i\) refers to one work task node representing one work task \(i\) that stakeholders need to participate in; The Object \(\text{WorkTask}_{\text{Auto}}_n\) refers to one work task node representing a work task \(n\) which a computer does automatically; Each Relationship\(^4\) \(\text{Seq}_m\) is one edge between nodes which represents sequence flow \(m\); Object \(\text{start}\) and \(\text{end}\) represent one start node and one end node, separately; Each Object \(\text{Gate}_k\) refers to one decision-making node \(k\) based on Gateway in BPMN [18].

Each information pattern formalizes system views of product architecture related to the corresponding work task, such as Objects requirement, function, model structure and V&V (The class diagram of the information pattern as one example is shown in Fig.3-A and B). The Objects model structure, function and V&V associated with the Object requirement. The Object model structure represents a simulation model structure. It includes Objects component and line representing blocks in the model and links between them. For example, when an Object model structure represents a Simulink model, its components and relationships represent blocks and lines in the model, if Matlab/Simulink is selected as a simulation tool for V&V. Object function describes functions of a simulation model and related components. The Object V&V associated with requirement contains Objects task, start, end, SimulationParameter, ModelParameter and ResultData to describe V&V activities. In V&V, each Object task is defined as one

---

\(^1\)The meta-modeling language, an extension of GOPPR is introduced in http://users.jyu.fi/~jpt/ME2000/Mc07id003.html.

\(^2\)Graph refers to a collection of object, relationship and role types represented as one window (one integrated concept of class diagram and package in UML).

\(^3\)Object refers to describe a thing with properties, relationships and roles, such as class in UML.

\(^4\)Relationship refers to one link between two Objects.
simulation execution with specific configurations according to the corresponding "Objects."

Definition 2: The information pattern related to one work task \((i|n)\) in the process \(a\) is defined as the Graph \(IM_{(a,i)}\).

\[
IM_{(a,i|n)} = WorkTaskDecomposite(WorkTask_Human_{i}|WorkTask_Auto_{a})
\]  

\[
IM = \text{Set}(\text{Set}(\text{Req}_{a,n}, n \geq 0), \text{Set}(\text{Func}_m, m \geq 0), \\
\text{Set}(s\text{Arc}_k, k \geq 0), \text{Set}(\text{V&V}_x, x \geq 0), \\
\text{Set}(\text{IMRelationship}_y, y \geq 0))
\]

where \(\text{WorkTaskDecomposite()}\) refers to a decomposition that one \(\text{Object WorkTask (WorkTask \_Human and WorkTask \_Auto)}\) decomposed into a new Graph \(IM_{(a,i)}\); \(\text{Req}_{a,n}\) is an Object requirement \(n\); \(\text{Func}_m\) refers to one Object function \(m\); \(\text{Object sArc}_{k}\) refers to a model structure \(k\). \(\text{V&V}_x\) refers to one Object \(\text{V&V}_x\). \(\text{IMRelationship}_y\) refers to relationships between previous \(\text{Objects in IM}\).

\[
s\text{Arc}_k = (\text{Set}(\text{Comp}_n, n \geq 0), \\
\text{Set}(\text{CompRel}_m, m \geq 0))
\]

where \(\text{Comp}_n\) and \(\text{CompRel}_m\) refer to the component \(n\) and line \(m\) between components in the model structure.

\[
\text{V&V} = (\text{Set}(\text{task}_{a,n}, n \geq 0), \\
\text{Set}(\text{RelationshipsInVV}_m, m \geq 0), \text{start}, \text{end}, \\
\text{Set}(\text{SMC}a, a \geq 0), \text{Set}(\text{SParam}_b, b \geq 0), \\
\text{Set}(\text{SRes}_c, c \geq 0))
\]

where \(\text{task}_{a,n}\) refers to one node representing one simulation with given configurations defined based on \(\text{SMC}_a\), \(\text{SParam}_b\), \(\text{SRes}_c\) which describe simulation configurations, parameter configurations and simulation results, separately. \(\text{RelationshipsInVV}_m\) refers to sequence and association relationships between nodes.

Definition 3: In the MBSE tool-chain, DSM models are transformed to services which can control tools and deploy technical resources, such as models. One simulation model related to \(s\text{Arc}_{k}\) is defined by \(\text{SM}(s\text{Arc}_{k})\). Each component and relationship in the \(\text{SM}(s\text{Arc}_{k})\) are defined as \(\text{SMCom}(\text{Comp}_n)\) and \(\text{SMRela}(\text{CompRel}_m)\), separately; During \(\text{V&V}\), one simulation execution \(\text{task}_{a,n}\) is defined by \(\text{SRun}(\text{task}_{a,n})\); The simulation configuration \(a\) for \(\text{SRun}(\text{task}_{a,n})\) is defined by \(\text{sCon}(\text{SRun}(\text{task}_{a,n}), a)\); The simulation parameter configuration \(b\) for \(\text{SRun}(\text{task}_{a,n})\) is defined by \(\text{SParam}(\text{SRun}(\text{task}_{a,n}), b)\); The related simulation result \(c\) is defined by \(\text{SRes}(\text{SRun}(\text{task}_{a,n}), c)\); A mapping between DSM models and technical resources is defined as follows:

\[
a = \text{ResMapping}(b)
\]

where \(\text{ResMapping()}\) refers to a mapping from one DSM model concept \((b=s\text{Arc}_{k}...S\text{Res}_c)\) or the related ontology concepts in DSM models to one technical resource \(a\) \((a=\text{SM}(s\text{Arc}_{k})...\text{SRes}(\text{SRun}(\text{task}_{a,n}), c))\). For example, \(\text{SM}(s\text{Arc}_{k})=\text{ResMapping}(s\text{Arc}_{k})\) where \(\text{ResMapping()}\) can be implemented by the property \(\text{modelName}\) as a tag to describe the real SM it represents.

Definition 4: One tool operation is defined as an execution \(\alpha\) of related tools to deploy the previous technical resources and support configurations:

\[
\text{toolOper}(\alpha, \text{tool}, \text{techResource}, \text{constraint}, \\
\text{APItemplate}, \text{mode})
\]

where \(\text{tool}\) refers to the related tool supporting such tool operations. \(\text{techResource}\) refers to technical resources used for such tool operations, such as \(s\text{Arc}_{k}\)\(, \text{constraint}\) refers to constraints of the tool operations, e.g. during creating one model, one block from the original model should be copied to a target path which the original block and the target path refer to as the \(\text{techResource}\) and \(\text{constraint}\), separately. \(\text{APItemplate}\) refers to a set of APIs supporting the related operations; \(\text{mode}\) refers to a trigger of the related tool operation, such as executing manually or automatically.

D. SPIRIT Framework Supporting Tool integration

As shown in Fig.2, after DSM models are built, ontology is generated through a developed code generator. Then the related concepts in ontologies are transformed to a WPMS and related services of deployment of technical resources and information of system artifacts. Such services, constructing a service infrastructure, wrap technical resources and ontology into unified forms which can be accessed through URL. Moreover, a service orchestration is developed to link related work tasks in the WPMS and required services of technical resources and information in ontologies. The related concepts are defined as follows:

Definition 5: An ontology concept, \(\text{Ong}(x)\) refers to one concept mirrored from the \(x\) concept in \(\text{PM}\) and \(\text{IM}\).

Definition 6: A service concept, \(\text{Ser}_n(x)\) is a web service \(n\) offered by service adapters for ontology and technical resources, which \(x\) refers to \(\text{Ong}(x)\) and technical resources, such as \(\text{SM}\) and \(\text{toolOper}\).

Definition 7: A service provider \(i\) for one work task, \(\text{SerProvider}_i = \text{Set}(\text{Ser}_n(x), n \geq 0)\) is a set of services in each work task \(i\).

Definition 8: \(\text{wProcess}_{a}\) refers to one process \(a\) in WPMS. \(\text{wTask\_Human}_{a}(i)\) and \(\text{wTask\_Auto}_{a}(n)\) refer to work task \(i\) for human and work task \(n\) implementing automatically in \(\text{wProcess}_{a}\).

Definition 9: Notation \(a\rightarrow b\) refers to one transformation between \(a\) and \(b\) which represent concepts in DSM models, ontologies, services and WPMS. Notation \(a) > b\) refers to one link from \(a\) to \(b\) which represent concepts in services, service providers and WPMS.

In Algorithm 1, an approach is presented to generate \(\text{wProcess}_a\) from DSM models. Firstly, DSM models including \(\text{PM}_a\) and \(\text{IM}_{(a,i)}\) are transformed to the corresponding ontology concepts using a developed code-generator. Then the ontology concepts related to \(\text{Ong}(\text{PM}_a)\) are used to generate a web-based process management system through a developed compiler. The each \(\text{Ong}(\text{IM}_{(a,i)})\) is transformed to the related \(\text{SerProvider}_i\) linked with \(\text{wTask\_Human}_{a}(i)\) in the \(\text{wProcess}_a\). Moreover, based on
ontologies, the SerProvider_i links with the relevant services of Ong(IM(a,i)) in each work task.

In addition, services of technical resources are linked with the corresponding services of Ong(IM(a,i)), such as Ser(SM(sArc_k)) and Ser(SRun(task_k)) are linked with Ser(sArc_k) and Ser(task_k), separately. As shown in TABLE I, in order to implement tool operations to support deployment of technical resources, each service of technical resources is linked with the related tool operation services. Finally, stakeholders can access the technical resources and implement related tool operations through such services.

### IV. Case Study

Based on the SPIRIT framework, we developed a toolchain prototype using the tools shown in Table II. The related workflow is shown in Fig.5:

1. The meta-models developed in MetaEdit+, are used to build DSM models to formalizing the domain-specific views of CPS development. Then through a developed code generator, ontologies in XML file are generated.
2. XML is loaded by a developed compiler including a process modeling engine, several OSLC adapters and a service orchestration.
3. A WPMS is generated by the developed compiler.
4. Stakeholders use their personal work tasks in the WPMS to implement their modeling work through related OSLC services.

The case study also demonstrates a process to build a Simulink model and execute one simulation by the tool-chain. Two main work tasks are as follows:

- **Worktask1** refers to one work task to create a model with a *Sin source block* connecting to a *Scope* block in Simulink.
- **Worktask2** refers to verification&validation work task for model configurations and parameter settings, simulation executions and browsing the simulation result on one stakeholder’s personal page in the WPMS.

When the WPMS is generated, each wTask_Human(a,i) is linked with the related SerProvider_i. In detail, in wTask_Human(a,i) (Worktask1), Ser(Req_i), Ser(Func_i), Ser(sArc_i) are linked through SerProvider_i. The Ser(sArc_i) are linked with Ser(SMCom(sArc_i)). Moreover, Ser(CompScope) and Ser(CompReline) in Ser(sArc_i) are linked with Ser(SMCom(CompScope)), Ser(SMCom(CompReline)) and Ser(SMCom(CompReline)), separately. Stakeholders can access the related technical resources through their own URL. They can implement related tool operations to create models, such as 'create_model' and 'add_block' etc. In wTask_Human(a,2) (Worktask2), one developer can configure the Simulink model and execute the simulations through the related OSLC services. He/she can also browse the related simulation results in the WPMS.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-specific modeling</td>
<td>MetaEdit+</td>
</tr>
<tr>
<td>Ontology description</td>
<td>XML</td>
</tr>
<tr>
<td>Process modeling engine</td>
<td>BPM Camunda</td>
</tr>
<tr>
<td>Service-based infrastructure</td>
<td>Related techniques of OSLC</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Matlab/Simulink</td>
</tr>
</tbody>
</table>

| TABLE II TOOLS USED IN THE TOOL-CHAIN Prototype |
|-----------------------------------------------|------------------------|
| **Techniques**                               | **Tools**              |
| Domain-specific modeling                     | MetaEdit+              |
| Ontology description                         | XML                    |
| Process modeling engine                      | BPM Camunda            |
| Service-based infrastructure                  | Related techniques of OSLC         |
| V&V                                           | Matlab/Simulink        |

V. Discussion

From the case study, we find the MBSE tool-chain prototype can support Matlab/Simulink executing simulations automatically. The stakeholders’ domain-specific views are described by the DSM models built based on meta-models. Then the DSM models are transformed to a WPMS which controls Matlab/Simulink and deploys related models automatically.
Through the service-oriented approach based on OSLC, technical resources - such as Simulink blocks, models and APIs - are wrapped into web-based services which promote their interoperability.

In our previous work, we have summarized two improvements of adopting a service-oriented approach [21]:

1. Improve **tool-integration**: Using service-oriented approach, data and models in different tools are represented as RESTful services in order to be accessed by other tools. APIs in tools are represented as services which makes tools can be controlled in a unified platform. This leads to an improvement of tool interoperability and reusability.

2. Improve the "interrelationships" Management of technical resources: The services enables process integration between process management system and related technical resources which improves traceability and consistency between system development and implementations.

3. Practices of systems of systems (SOS). The **SPIRIT** framework is an exemplified SOS. Its related research provides engineers a industrial practice to understand SOS.

From our experiences, we note that there are still several limitations when the framework is used to implement the industrial practices for tool-chain development. Firstly, it is challenging to define a clear system boundary for the social network. Stakeholders should formalize their workflows (including development process and information) and define the related social networks based on systems engineering views. Secondly, tools for current CPS development are always specific and closed environments thus not enough APIs can be provided to support service development. Thirdly, interfaces between the MBSE tool-chain and existing PDM and PLM platforms should be defined. Open standards, such as OSLC can promote the interoperability between tool-chain and existing platforms.

From the technical view of the MBSE tool-chain, future work should be considered: DSM models can be formalized based on meta-models, however, except for the needs of domain-specific views, the meta-meta models (semantics and syntax) also need to be extended. Then further research would be done to improve ontology by considering open standards. Moreover, during service-orchestration, specifications of service (such as URI of each service) need to be further defined. In this paper, we adopted a sample case study to illustrate how the prototype supports Matlab/Simulink executing the simulations. Further V&V techniques, such as co-simulation would be extended in the future.

VI. Conclusion

In this paper, we proposed the **SPIRIT** framework to support MBSE tool-chain development. The framework integrates social networks of stakeholders, CPS development processes, system artifacts and technical factors, e.g. modeling methods and technical resources which can capture related requirements...
about the target tool-chain development. Moreover, based on the SPIRIT framework, a service-oriented approach is used for tool-chain development to promote interoperability. The tool-chain encompasses domain-specific modeling, process management based on IT platforms and standard based tool-integration. With the developed tool-chain prototype, CPS developers can formalize their products from a domain-specific view, deploy related technical resources with well-managed and good interoperability, and implement related verification & validation automatically.

REFERENCES


