A Roadmap Towards Integrated CPS Development Environments

Jad El-khoury, Fredrik Asplund, Matthias Biehl, Frederic Loiret and Martin Törngren
Mechatronics Lab, Department of Machine Design, Royal Institute of Technology, Stockholm, Sweden
{*jad, fasplund, biehl, floiret, martint*}*@kth.se*

Keywords: Tool Integration, Cyber-Physical Systems, Development Environment, Tool Chain, Interoperability Standards

Abstract: Cyber Physical System (CPS) development is highly heterogeneous, involving many stakeholders, each of which interacts with its development artifacts through a variety of tools, and within several engineering processes. Successful CPS development requires these tools to be well-integrated into a Development Environment (DE) in order to support its many stakeholders and processes. In this paper we identify the main challenges facing DE development for CPSs, and presents a roadmap to meet these challenges. We here take the position that focus should be redirected from trying to achieve a single, one-size-fits-all solution to such a heterogeneous problem. Instead, focus should be placed on supporting the development of highly-customized DEs, which readily can be applied to industrial development. Such a highly-customized DE should fit the needs of a particular development organization, while at the same time taking advantage of relevant standardization efforts.

1 INTRODUCTION

Cyber Physical Systems (CPSs) offer opportunities for new services, improved performance and better efficiency in almost all application domains in our society.

Typically many technical and business-related stakeholders take part in the development of a CPS. Each of these stakeholders interacts with the development artifacts through various tools, pertaining to their particular interests. Mechanical engineers for instance make use of CAD tools to construct artifacts that describe the mechanical aspects of the CPS under development. This interaction takes place within several technical engineering processes involving structured sequences of activities such as requirements engineering, design and verification.

Due to the tightly integrated technologies in a CPS, all of these technical engineering processes become tightly intertwined and decisions made by one stakeholder become likely to have an impact on other stakeholders. An effect of this is that the tools supporting each separate technical engineering process needs to be adequately integrated with tools of other technical engineering processes. This becomes problematic, since many of the tools employed throughout the different processes typically come from separate sources and are hence likely to be mutually incompatible.

This paper focuses on the overall problem aimed at supporting the integration of these mutually incompatible tools. We will use the term Development Environment (DE) to refer to a setting of tools that support multiple stakeholders and processes in CPS development. How to design and maintain DEs have been discussed thoroughly these last three decades, with the overall focus being on trying to achieve a single, one-size-fits-all solution towards which all mutually incompatible tool technologies subsequently can interface (thereby forming a homogeneous unity).

We believe that the current approaches to building DEs are built on a false hope that CPS development can be homogenized, so that a single homogeneous DE integration framework to support the whole CPS development life-cycle can be provided. As we will argue in this paper, CPS development is too heterogeneous to allow for such a single homogeneous solution. Instead, we accept the heterogeneous nature of CPSs and their DEs, and aim to instead provide well-integrated heterogeneous DEs.

1.1 Position Statement

Given the heterogeneous nature of CPS development, and the (current) slow pace of convergence of integration technologies, we believe that focus should be redirected at supporting the development of highly-customized and maintainable DEs, which readily can be applied to industrial development. Such a highly-customized DE should fit the needs of a particular development organization, while at the same time taking advantage of relevant standardization efforts.

A tailorable organization-specific DE can only be practically feasible if the threshold of its development is lowered by providing DE development and automation support. Moreover, a concerted effort will be required to provide the required methodology, standards and business models required for large scale industrial adoption of efficient tool integration.

1.2 Paper Structure

This paper first describes in section 2 the heterogeneous context of CPS development, leading to the main challenges of developing DEs - which are elaborated in section 3. Section 4 finally summarizes the issues that need to be addressed to move forward and some of the opportunities that this will lead to.

2 THE HETEROGENEOUS CONTEXT OF CPS DEVELOPMENT

The CPS development process is highly heterogeneous. In this section we go through some of the more important aspects in which this has an impact on DEs.

2.1 Tool and Tool Integration Heterogeneity

As mentioned in section 1, many stakeholders with different technical and business-related specialities mean that many heterogeneous tools will be found in a DE. The tightly intertwined technologies in an CPS will then mean that many tools will be integrated with each other, forming complex dependencies between them.

A DE is often built in a bottom-up manner, eventually displaying an unstructured design and implementation of tool integration. Such ad-hoc realizations of DEs may use a variety of integration frameworks, data formats, communication protocols and assumptions. Integration conventions provide a common ground for building DEs and increase the likelihood that parts of DEs can be reused in a different context than they were originally designed for. However, several conventions for integration exist, such as XMI (XML Metadata Interchange) (OMG, 2007), OSLC (Open Services for Lifecycle Collaboration) (OSLC Core Specification Workgroup, 2010) and STEP (ISO, 1994). Similarly, several specialized technologies for realizing the different parts of a DE are currently available, such as model transformation tools, tracing tools or libraries for exposing services of tools. Each of these integration technologies describes only one aspect of the DE, while a complete DE needs to cover several aspects.

This plethora of tools, integration frameworks, integration conventions, languages and technologies for realizing parts of a DE, combined with the common ad-hoc realization approach, typically lead to a rich heterogeneity of technologies used for tool integration.

2.2 Organizational Heterogeneity

Organizations have different development processes of varying technical maturity. For example, while a more traditional DE supports simple connections between a small number of tools; a modern DE may need to support development processes that are model-based and iterative (Tratt, 2005) and include a larger number of tools.

In addition, to get a DE accepted, it is important that its end-users (such as the different types of engineers, architects, managers, designers, analysts, etc.) are involved (Christie et al., 1997). DEs therefore need to be tailored to each specific organization, or even each specific development project. As a consequence, the requirements and nature of tool integration vary among organizations.

2.3 Stakeholder Heterogeneity

The DE end-users frequently see the ideal case as being when they can focus on one tool to support a particular ”main” activity and then have information automatically flow to and from this tool (Maalej, 2009). However, DE users are not the only stakeholders relevant to tool integration.

DE designers, deployers and maintainers have a different view of each tool and the overall DE. They instead usually favor tampering as little as possible with each tool or technology employed for tool integration.

And while those stakeholders have a common ground in the focus on technology, other stakeholders have an all together different focus. Management commonly looks at the *cost* of procuring technologies and tools when deciding which solution to favor. Avoiding a potentially costly ”lock-in” in regard to a particular technology can lead to the rejection of tools that are technologically superior.

This focus on economical factors can also be found in the reasoning of tool vendors, which may be interested in tool integration as an argument for customers to choose their tools (for instance to increase the odds of avoiding a costly "lock-in"). However, tool vendors with market unique or dominating solutions have less of a reason to offer support for much tool integration technologies. This support comes with a cost, both during development and maintenance, which a tool vendor might want to avoid since it is not their main business.

3 THE CHALLENGES OF DE DEVELOPMENT FOR CPS

Given the heterogeneous nature of CPS development, we here identify and elaborate on the most important challenges facing DE development.

3.1 Lack of Support Methodologies and Tools

A DE consists of many distributed software assets to be integrated with each other. This is complicated due to the heterogeneity described in section 2, and the many intricate dependencies mentioned in section 1. Several barriers have to be bridged, including *technology* (e.g. technical representation of information and functionalities), *semantics* (meaning of information and functionalities, and their relations), and *intended interactions* (scenarios involving two or more tools).

Part of the problem is the lack of an established methodology for the development of DEs, meaning that DEs are often implemented in an ad-hoc manner. This leads to "fragile integrations" (Derler et al., 2012) that are difficult to extend and maintain.

Another level of complexity is introduced when needs for customization have to be considered, e.g., to take into account product-specific lifecycle features and integration patterns.

Current platforms for tool integration provide partial solutions, but also introduce accidental complexity through the amount of manual coding, low-level technologies and configurations. The lack of support methodologies comes along with a corresponding lack of support tools that would offload the burden on DE developers and integrators at various stages of the development process. For instance, high level modeling languages for designing and supporting early testing of DEs, coupled with adaptive composition mechanisms and automatic synthesis of integration assets, are typical support tools that are not available to enrich the portfolio of DE developers and integrators.

3.2 Convergence towards New Standards for Tool Integration

Supporting methodologies and tools needs to be complemented by suitable standards. The question is, however, how come there is so few widely adopted standards for tool integration, when there is clearly no lack of suggestions for how to achieve standardization? There is for instance a multitude of suggestions for data exchange formats (such as XMI (OMG, 2013b), FMI (FMI Development Group, 2013) and ReqIF (OMG, 2013a)), modeling languages (such as EAST-ADL (EAST-ADL Association, 2013)) and even complete frameworks (Such as Jazz (IBM, 2013) and ModelBus (ModelBus Team, 2013)) to support one or more aspects of tool integration.

It could have been expected that - over time - momentum would have been picked up behind a selected few of these diverse suggestions, leading to a more concentrated effort towards a settled set of basic standards. Time and experience is after all needed for this kind of maturity process, given the many stakeholders (and their complex relationships) involved in any such efforts. While this process is natural for any emerging technology, the heterogeneous context of CPS development (as presented in section 2) unfortunately causes a prolongation of the time period until such stable standards are in place - if at all.

Furthermore, as mentioned in section 2, tool integration is a secondary objective for many of the key stakeholders. For example, CPS developers and the business units they belong to want to develop a product, while tool vendors want to develop tools with minimum effort and sometimes believe that open tools is a threat rather than opportunity, etc. In the best case, business units can cooperate with such vendors to solve their particular integration needs, but it leads to organization-specific (and in many cases ad-hoc) integration solutions. This obviously further prolongs the time until standards pick up enough momentum to become widely accepted.

Figure 1 illustrates the cyclic nature of the challenge in engaging the key stakeholders through constructive interactions in order to converge on integration technologies. The lack of support methodologies and tools contribute to the vicious nature of the cycle.

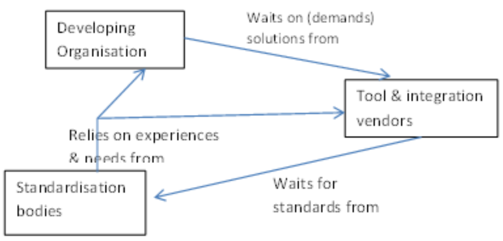
Developing Organisation
Waits on (demands) solutions from
Tool & integration vendors
Relies on experiences & needs from
Standardisation bodies
Waits for standards from

Figure 1: key stakeholders integration technologies and their dependencies.

4 CONCLUDING DISCUSSION

We believe there is a need for a new approach in order to break the vicious circle illustrated in Figure 1. As mentioned in section 3, the key stakeholders lack the incentive to push for a solution to such a need, given their own primary objectives.

A very important aspect is to raise the level of maturity and awareness of the developing organizations in order for them to be able to take a more active role in contributing to workable standards.

Another important aspect is to establish integration methodologies and standards that take the heterogeneity and needs for customization explicitly into account. This will enable tool vendors to provide customized integration solutions that can be relatively easily and cheaply provided for any developing organization.

Finally, there is an opportunity for additional business players to provide standardized tool interfaces for existing development tools and use these interfaces to create customized and automated development environments. We call these business players *integration providers*. A business model for such a business player would take advantage of the current vacuum to provide customized, standardized tool integration, allowing these players to act as product or service providers (See (Murphy and Duggan, 2012) for example).

Maturing methodologies and standards will strengthen the opportunities of integration providers, promising to create a successful industrial ecosystem. Getting there will require both research and collaborative efforts among key stakeholders.

REFERENCES

Christie, A., Levine, L., Morris, E. J., Riddle, B., and Zubrow, D. (1997). Software Process Automation: Interviews, Survey, and Workshop Results. Technical report, SEI.

Derler, P., Lee, E. A., and Vincentelli, A. S. (2012). Modeling Cyber-Physical Systems. *Proceedings of the IEEE*, 100(1):13–28.

EAST-ADL Association (2013). East-adl.

FMI Development Group (2013). Functional mock-up interface.

IBM (2013). Ibm rational jazz.

ISO (1994). Industrial automation systems and integration – product data representation and exchange (ISO 10303). Technical report, ISO.

Maalej, W. (2009). Task-First or Context-First? Tool Integration Revisited. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, pages 344–355, Washington, DC, USA. IEEE Computer Society.

ModelBus Team (2013). Modelbus.

Murphy, T. E. and Duggan, J. (2012). Magic quadrant for application life cycle management. Technical report, Gartner.

OMG (2007). MOF 2.0 / XMI Mapping Specification, v2.1.1. Technical report, OMG.

OMG (2013a). Requirements interchange format (reqif).

OMG (2013b). Xml metadata interchange.

OSLC Core Specification Workgroup (2010). OSLC core specification version 2.0. Technical report, Open Services for Lifecycle Collaboration.

Tratt, L. (2005). Model transformations and tool integration. *Software and Systems Modeling*, 4(2):112–122.