Proceedings of the 1st Scandinavian Workshop on the Engineering of Systems-of-Systems (SWESoS 2015)

Kista, May 27 2015

Jakob Axelsson (Editor)
jakob.axelsson@sics.se

SICS Technical Report T2015:04

Contents

Preface

The rapid digitization of society is to a large extent driven by the interconnection of existing systems in order to co-ordinate their activities. This leads to systems-of-systems (SoS), where the parts more or less voluntarily co-operate for mutual benefits while keeping their autonomy. The term SoS started to become relevant some 20 years ago, and accelerated as a research area about 10 years ago. Although some people tend to take SoS as a synonym for large and complex systems, the research community has arrived at a fairly precise characterization of the term. In an SoS, the elements, or constituent systems, exhibit an operational and managerial independence, meaning that they can operate outside the SoS context, and have different owners. They choose to collaborate in order to achieve a common goal, manifested as an emergent property of the SoS, i.e. a property not existent in any of its parts in isolation.

The field so far has been dominated by US researchers focusing on military and space applications. Key topics include architecture, communications, interoperability, modeling and simulation, and also a number of properties where dependability attributes such as safety play an important role. From its origins in the government driven sectors, SoS are now spreading to civilian and commercial usage.

To investigate the needs and strategies for Sweden in relation to SoS, VINNOVA in late 2014 commissioned a consortium led by the Swedish Institute of Computer Science (SICS) to develop a research and innovation agenda for the area. The agenda project has included an industrial perspective captured in a series of workshops with practitioners, and also a research perspective. The latter was handled through an extensive research literature review, which indicated a poor representation of Scandinavia in the SoS research community. Also, a survey was sent to all relevant Swedish universities, research institutes, and funding agencies, and the result of this was somewhat contradictory. Many researchers are indeed working on topics related to SoS, but often use different terms for it, and publish at other venues than the SoS community.

Given the large, but scattered, activity in the highly multidisciplinary SoS area, SICS and the Swedish Chapter of the International Council on Systems Engineering (INCOSE) decided to organize the 1st Scandinavian Workshop on the Engineering of Systems of Systems (SWESoS 2015). The primary purpose of the workshop was to create a meeting place for researchers and practitioners interested in SoS. The workshop was intended to be an informal event, focusing on presentation of results and ongoing research, to stimulate interaction among the researchers. This proceedings volume contains the extended abstracts of those presentations. In many cases, the presentations are based on work already published elsewhere, and the interested reader can find links to more material in each contribution.

The scope of the workshop was all aspects related to SoS engineering. This included, but was not restricted to, the following topics when applied to systems of systems: Autonomous and cooperative systems; Business models, including software ecosystems; Case studies of applications in different domains; Control strategies; Communication; Dependability, robustness, and other quality attributes; Enterprise architecture; Governance; Interoperability; Modeling and simulation, including multi-agent systems; Service oriented architecture; Systems engineering methods; and Systems thinking.

In total, 16 papers were submitted to the workshop, and 13 were accepted for presentation, whereas the remaining three were somewhat outside the core scope of the event.

The ‘Three Layer Ecosystem Strategy Model’ (TeLESM)

Helena Holmström Olsson and Jan Bosch

Malmö University and Chalmers University of Technology

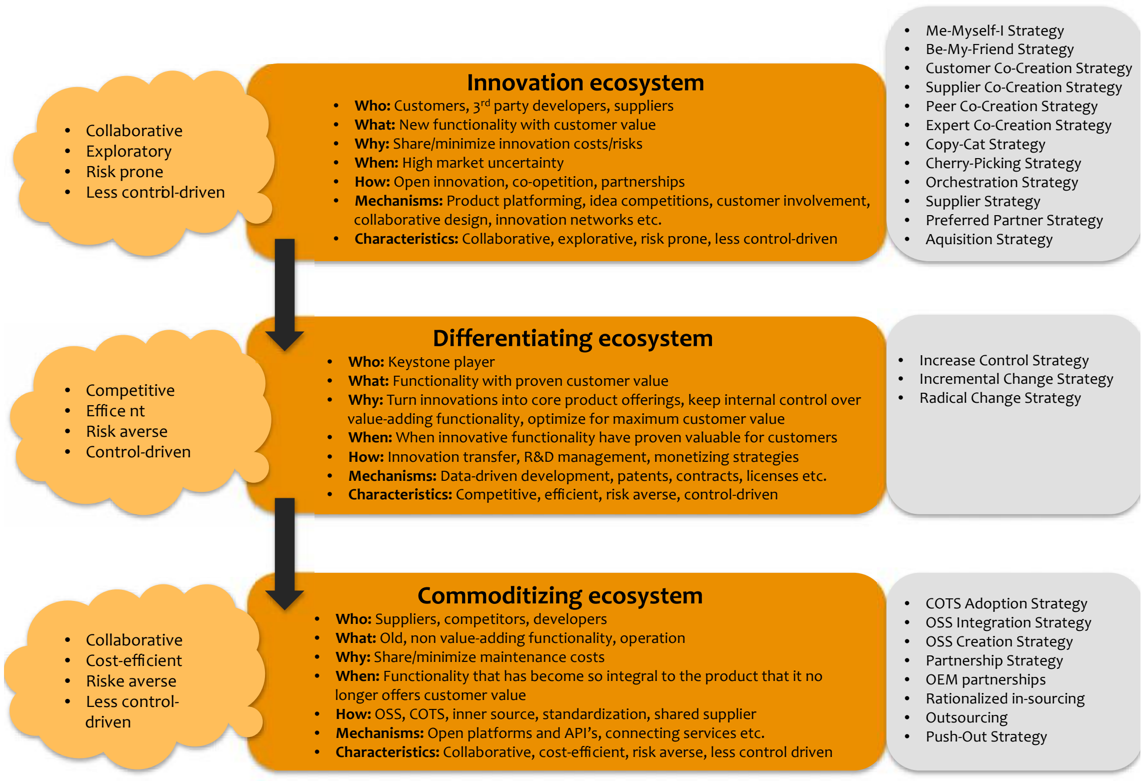
helena.holmstrom.olsson@mah.se, jan.bosch@chalmers.se

Recently, business ecosystems have gained significant attention in the software engineering research community. The concept refers to the shift from an intra-organizational perspective to an inter-organizational perspective where product development and innovation is moving out from the organizational boundaries and where networks of stakeholders co-create value (Janssen et al 2012; Scacchi and Alspaugh 2012; Ritala et al 2013; Dagnino and Padula 2002). As defined by Moore (1993; 1996), a business ecosystem includes suppliers, lead producers, competitors, and a number of other stakeholders that over time co-evolve their capabilities and roles, and align themselves with the directions set by one or more key stakeholders.

To operate successfully in a business ecosystem, companies need strategies that help them move away from ad hoc interventions with external stakeholders, to instead adopt a strategic approach on how to manage relationships and dependencies to external forces and interests. As recognized in a number of studies, the increasing interdependencies between stakeholders and organisational networks, introduces technical as well as managerial challenges (Olsson and Bosch 2014; Messerschmitt and Szyperski 2003; Bosch-Sijtsema and Bosch 2014; Santos et al 2012). From a technical perspective, challenges typically involve technical infrastructures and architectures that allow for easy integration of third party content (Santos et al 2012). From a managerial perspective, challenges involve e.g. coordination of standardization efforts, sharing of maintenance costs and how and when to engage in open innovation initiatives etc.

While there is research focusing on the managerial perspective of ecosystems, (Olsson and Bosch 2014; Van den Berk et al 2010; Bosch 2012), few studies provide guidance that helps companies distinguish between the multiple ecosystems they operate in, and for which companies need different strategies for achieving their goals.

Therefore, and to address this gap, we develop the ‘Three Layer Ecosystem Strategy Model’ (TeLESM) presented in Figure 1. As a theoretical foundation, we use the ‘Three Layer Product Model’ (3LPM) that was developed to help companies distinguish between distinct layers of system functionality in order to reduce architectural complexity (Bosch 2013). While the purpose of this model is to help solve architectural complexity, it recognizes the importance of distinguishing between different functionality layers. In our view, this distinction is equally important in order to understand the multiple ecosystems that companies operate in, and the different strategies they apply to manage these. In the development of our model, we translate the 3LPM layers into the (1) innovation ecosystem, (2) the differentiating functionality ecosystem, and (3) the commoditizing functionality ecosystem, and we argue that companies need to deploy different sets of strategies to successfully manage these ecosystems. In our model, the *innovation ecosystem* is about hypothesizing about future differentiating functionality through customer interaction and analysis of competing products, as well as about brainstorming and other idea creation initiatives. The *differentiating ecosystem* is about identifying and incorporating new functionality as it develops in the innovation ecosystem, i.e. recognizing and transferring functionality that has proven valuable and appropriately transitioning it to the lower layer. Finally, the *commoditized ecosystem* is about incorporating as much functionality from the differentiating layer into the commoditized layer as feasible in order to replace proprietary software with commercial software components whenever possible.



Collaborative
Exploratory
Risk prone
Less control-driven
Innovation ecosystem
Who: Customers, 3rd party developers, suppliers
What: New functionality with customer value
Why: Share/minimize innovation costs/risks
When: High market uncertainty
How: Open innovation, co-opetition, partnerships
Mechanisms: Product platforming, idea competitions, customer involvement, collaborative design, innovation networks etc.
Characteristics: Collaborative, explorative, risk prone, less control-driven
Me-Myself-I Strategy
Be-My-Friend Strategy
Customer Co-Creation Strategy
Supplier Co-Creation Strategy
Peer Co-Creation Strategy
Expert Co-Creation Strategy
Copy-Cat Strategy
Cherry-Picking Strategy
Orchestration Strategy
Supplier Strategy
Preferred Partner Strategy
Aquisition Strategy
Competitive
Effice nt
Risk averse
Control-driven
Differentiating ecosystem
Who: Keystone player
What: Functionality with proven customer value
Why: Turn innovations into core product offerings, keep internal control over value-adding functionality, optimize for maximum customer value
When: When innovative functionality have proven valuable for customers
How: Innovation transfer, R&D management, monetizing strategies
Mechanisms: Data-driven development, patents, contracts, licenses etc.
Characteristics: Competitive, efficient, risk averse, control-driven
Increase Control Strategy
Incremental Change Strategy
Radical Change Strategy
Collaborative
Cost-efficient
Riske averse
Less control-driven
Commoditizing ecosystem
Who: Suppliers, competitors, developers
What: Old, non value-adding functionality, operation
Why: Share/minimize maintenance costs
When: Functionality that has become so integral to the product that it no longer offers customer value
How: OSS, COTS, inner source, standardization, shared supplier
Mechanisms: Open platforms and API's, connecting services etc.
Characteristics: Collaborative, cost-efficient, risk averse, less control driven
COTS Adoption Strategy
OSS Integration Strategy
OSS Creation Strategy
Partnership Strategy
OEM partnerships
Rationalized in-sourcing
Outsourcing
Push-Out Strategy

Figure 1. The Three Layer Ecosystem Strategy Model (TeLESM).

The model was developed and validated based on case study research (Walsham 2006) conducted in six companies in the B2B software intensive systems industry. Our study included a total of 51 people that we met with during two rounds of interviews. In addition, we organized a number of workshop sessions at which representatives from all six case companies attended. As an outcome of our empirical research, the model defines the roles, the drivers, the purpose and the characteristics for each ecosystem layer. In doing this, our model helps companies address the inherent complexity of operating in multiple ecosystems, and it emphasizes the importance of selecting appropriate strategies to manage these. During the validation of the model, we re-visited the companies and we explored further the (1) completeness of the model, i.e. do companies employ strategies that are not part of the model, and (2) relevance, i.e. do companies use all strategies in practice. Our results show that the model provides an accurate framework for strategy selection, and that all case companies use a mix of different strategies to manage the multiple ecosystems they operate in.

In future research, we seek to improve the decision support for the timing of functionality transfer between the innovation and differentiating ecosystem, as well as between the differentiating and commoditizing ecosystem. Finally, we plan further validation of the TeLESM model to improve companies' capacity in realizing the potential of the multiple ecosystems they operate in.

References

Bosch, J. (2013). Achieving Simplicity with the Three-Layer Product Model, IEEE Computer, Vol. 46 (11), pp. 34-39.

Bosch J., and Bosch-Sijtsema P. (2014). Aligning innovation ecosystem strategies with internal R&D. In Proceedings of the 7th International Conference on Management of Innovation & Technology, September 23-25, Singapore.

Dagnino, G.B., and Padula, G. (2002). Coopetition strategy: towards a new kind of interfirm dynamics for value creation, EURAM 2nd annual conference, Stockholm School of Entrepreneurship, Sweden 8–10 May.

Jansen, S., Brinkkemper S., Souer J., and Luinenburg L. (2012). Shades of gray: opening up a software producing organisation with the open software enterprise model. The Journal of Systems and Software, Vol 85, pp. 1495-1510.

Messerschmitt, D.G., and Szyperski, C. (2003). Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press, Cambridge, MA, USA.

Moore, J. F. (1993). Predators and Prey: A New Ecology of Competition. Harvard Business Review.

Moore, J. F. (1996). The Death of Competition: Leadership & Strategy in the Age of Business Ecosystems. New York: HarperBusiness. ISBN 0-88730-850-3.

Olsson H. H., and Bosch J. (2014). Ecosystem-driven software development: A case study on the emerging challenges in inter-organisational R&D. In Proceedings of the 5th International Conference on Software Business, June 16-18, Paphos, Cyprus.

Ritala P., Agouridas V., Assimakopoulos D., Gies O. (2013). Value cration and capture mechanisms in innovation ecosystems: a comparative study. International Journal of Technology Management, Vol. 63, No. 3/4.

Santos R., Werner C., Barbosa O., and Alves C. (2012). Software ecosystems: Trends and impacts on software engineering. In Proceedings of the Brazilian Symposium on Software Engineering (SBES), pp. 206-210, September 23-28, Brazil.

Scacchi W., and Alspaugh T.A. (2012). Understanding the role of licenses and evolution in open source architecture software ecosystems. The Journal of Systems and Software, Vol. 85, pp. 1479-1494.

Van den Berk I., Jansen S., and Luinenburg L. (2010). Software ecosystems: A software ecosystem strategy assessment model. In Proceedings of the 4th European Conference on Software Architecture. Companion Volume, ACM, New York, NY, USA, pp. 127-134.

Walsham, G. (2006). Doing Interpretive Reseach. European Journal of Information Systems, (15), pp. 320-330.

Openness in the Interplay between Technical and Business aspects – a system of systems

Per Runeson

Lund University

per.runeson@cs.lth.se

Technical systems never exist in isolation, but interplay with users as well as other technical systems, and exist in a specific business context. With the arrival of various variants of open innovation, i.e. "a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as they look to advance their technology" (Chesbrough, 2003), the interplay between different technical and business systems becomes apparent. This interplay is not a hierarchical relation, but rather of network type, where some nodes may be more powerful than others (technically or business-wise), but there is a mutual dependency between the nodes. Further, the technical and business aspects are intertwined, composing a system of systems.

Software is a specifically well suited enabler for open innovation, since the mechanisms of open source software may foster an instance of open innovation, and that the intangible characteristic of software opens up for novel methods for product and service design and delivery (Regnell et al. 2015, Munir et al. 2015). Business models for software based services and products also vary a lot, from the traditional paid license model – Microsoft being the role model – via other companies distributing free software at a cost for related services – for example RedHat– to yet other companies getting paid by advertising as a return for the information users provide for free – read Google. Depending on the business model, the technical architecture and working practices vary.

We have studied some aspects of the interplay between the technical and business systems through surveys and in observational case studies, where companies open up their business and technical systems, and we clearly show dependencies between the systems as well as the lack of knowledge and guidance for strategic and operational decisions. We also identify interdependencies between different types of innovation, product, process, business and organizational innovation (Linåker et al, 2015).

Axis is a supplier of networked surveillance cameras. They collaborate with integrators to supply complete, tailored systems for the end users. Earlier, the integration was hierarchical, plugging cameras together into a complete system. Now, as they gradually move their features into software, the cameras must open up for execution of third party applications in the cameras. In order to boost open innovation, Axis has invited partners to a semi-open ecosystem, which we have studied (Wnuk et al, 2014). Our observations suggest that internal and external standardization can play a dual role, not only ease the development but also enable additional sales channels and new opportunities for the ecosystem participants. At the same time, the business model selected by the ecosystem leaders and technical execution performance, are identified as the main barriers to ecosystem participation.

Gerrit and Jenkins are open source tools that are used in Java system development. Gerrit is an OSS code review tool created by Google in connection with Android in 2007. It is tightly integrated with the software configuration management tool GIT, working as a gatekeeper. Jenkins is an open source build server that conducts the build of executable software from source files. The tools are integrated into a tool chain, constituting a technical system of systems, but there is also an interplay between technical and business systems. Both tools are developed and evolved through investments from companies. Consequently, companies have to take into consideration what to share openly, and what to maintain internally, i.e. the interplay between technical and business systems, at the technical design level. We observe that these trade-offs are made, but also that strategic support for management in making these decisions are lacking.

In summary, with open innovation, several challenges appear, both when integrating technical system from other systems, and with respect to the interplay between the technical and business aspects of products and services, based on software.

References

H. W. Chesbrough. *Open innovation: the new imperative for creating and profiting from technology*. Harvard Business School Press, Boston, Mass., 2003.

J. Linåker, H. Munir, P. Runeson, B. Regnell, and C. Schrewelius. A survey on the perception of innovation in a large software organization. In K. Wnuk and R. J. Machado, editors, *International Conference on Software Business*, 2015.

H. Munir, K. Wnuk, and P. Runeson. Open innovation in software engineering: A systematic mapping study. *Empirical Software Engineering*, online http://dx.doi.org/10.1007/s10664-015-9380-x, 2015.

B. Regnell, P. Runeson, M. Höst, and J. Linåker. Innovation med öppen källkod ger konkurrensfördelar. *Management of Innovation and Technology*, volume 2, May 2015.

K. Wnuk, P. Runeson, M. Lantz, and O. Weijden. Bridges and barriers to software ecosystem participation - a case study. *Information and Software Technology*, 56(11):1493–1507, 2014.

Frameworks for innovation – studies of technology innovation and systems for energy, defense and security

Bengt A Mölleryd

KTH

bamolleryd@gmail.com

This study concerns *technology innovation and systems for energy, defence and security*. The study focus on ways and means of *promoting and conditions for innovation* or value adding new technologies, systems, services and products for the mentioned areas and purposes.

Attention is particularly paid to innovation as phenomena which emerge from organizing *integrated system of systems* by *evolutionary processes* with vital elements of experimenting and learning.

Focus is on radical technological innovations that have disruptive scope and consequences. *Disruptive innovations* are commonly induced and driven by technology and technological changes. Digitization with associated net technologies is a prime example of technologies that cause major disruptions of systems, industrial branches and firms, and whole sectors of society. Radical or paradigmatic innovations contrast to "normal innovations" which come from minor changes of processes, mere re-designs or changed fashions and images of the existing products and services to the customers and users.

The study is aiming at an *innovation view*, which is a rudimentary *model and architectural framework* with a purpose to provide (strategic) guidance and means to governance of technological innovation. To the benefit for application and practice when designing and engineering innovation and innovating systems, the innovation view aligns with, but does not substitute recognised international enterprise architectural frameworks, systems engineering standards and protocols (for example ISO 15288, ISO 42010, NISP).

Cases of technological innovation and systems for energy, defence and security are studied with regards to the proposed innovation view and relevant architectural frameworks.

Keywords: disruptive innovation, technology and innovation, systems integration, systems of systems, energy, defence, security, security of supply, standards and protocols, evolutionary development, systems engineering, enterprise architectures and frameworks

References

Frameworks for innovation – studies of technological innovation and systems for energy, defence and security, 2015, Thesis draft, Department of Energy, ITM, KTH (not yet published).

Enterprise Architecture Modeling and Analysis of Quality Attributes – The Multi-Attribute Prediction Language (MAPL)

Mathias Ekstedt, Pontus Johnson, and Robert Lagerström

Industrial Information and Control Systems, KTH Royal Institute of Technology

{mathias.ekstedt, pontus.johnson, robert.lagerstrom}@ics.kth.se

Problem

Enterprise architecture is often considered to be a tool for making good decisions regarding the enterprise information systems and recently also other complex systems-of-systems (like the smart grid). Decision-making can be viewed as a process of scenario selection. From the current state of the system, various change decisions will result in new systems. Generally, decision makers have some ideas about how these scenarios could manifest themselves in the short and in the long term. Enterprise architecture models can represent each future scenario, as well as the current state. Models over for instance applications, business processes, information, and technical infrastructure may all be employed for specifying these scenarios. The main problem in decision-making is to choose which one of the future scenarios to pursue; which one is the better one for a given purpose.

The main difference between enterprise architecture and alternative approaches to complex systems management is perhaps the focus on models of the systems and the context within which they reside. There are other approaches to systems management that share many of the views of the enterprise architecture community, but no other approach places quite as large emphasis on modeling. So, what is a model? Common models familiar to most people include geographical maps, architectural drawings, and miniature buildings. As indicated by the term enterprise architecture, we are mainly concerned with graphical models, i.e. drawings over how various things and phenomena are related. The analogy between enterprise architecture and the traditional architecture of buildings is in many ways appropriate.

Models are powerful tools for mainly two reasons: Firstly, they help us focus on the important issues when contemplating a certain problem. A common map depicting the different nation states of the world leads us to focus on questions like what the capital of this or that country is or to which country some particular island belong, while a road map leads us to questions about the driving distance between various locations. A weather map, of course, leads us to other questions. Secondly, models provide different people with a common view of an issue. Models both provide a common language that helps us communicate with each other and also guide us to focus on the same set of issues. Models are thus effective tools for planning, communicating, and of course, also for documenting (remembering). It is thus an important mission of enterprise architecture to provide useful models for the various decision-making activities of enterprise information systems planning.

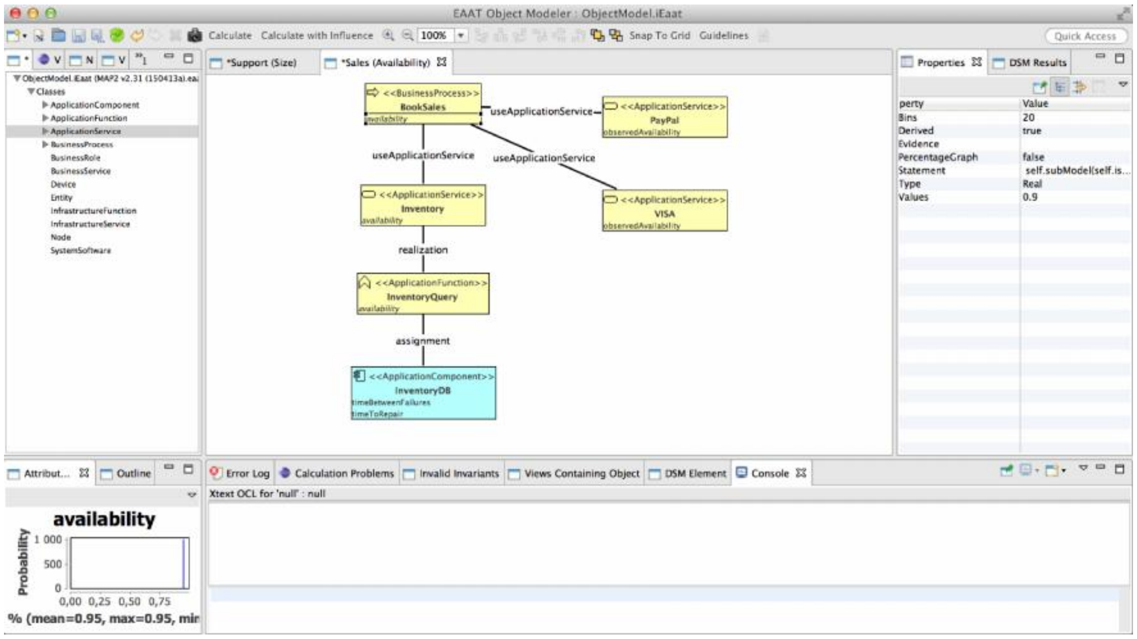
Modeling, however, can be costly. The world is full of things that could be represented, and it would not be difficult to spend completely unreasonable efforts on modeling the details of existing and future systems. Such indiscriminate modeling would be of little value, not only due to the effort of producing the models, but also because the models would soon be as difficult to understand as the real world they represent. In order to avoid indiscriminate modeling, we advocate for a goal-driven approach. In brief, only those phenomena that directly relate to our enterprise architecture goals are to be modeled. In other words, only the information required for answering our most pertinent questions will be gathered in the enterprise architecture models.

Our solution

We have developed a modeling and analysis tool with accompanying metamodels (modeling languages). These modeling languages define what a modeler needs to model in order to conduct the analysis wanted, in order to get decision support for scenario selection. The tool is called the KTH Enterprise Architecture Analysis Tool (EAAT[1]) and our most all-encompassing metamodel is called the Multi-Attribute Prediction Language (MAPL). MAPL have gone through many iterations [1-3] and is also stemming from research on how to conduct modeling and analysis of specific quality attributes, such as modifiability [4], data accuracy [5], and availability [6]. Today MAPL contains analyses for cost, coupling, size, availability, data accuracy, as well as utility theory that combine goals and requirements to make an overall architecture assessment for the trade-off between the quality attributes.

The MAPL is itself written in the Predictive, Probabilistic Architecture Modeling Framework (P^2AMF) [7]. This framework is in turn based on UML[2] and OCL[3], but extended with a feature of probabilistic inference (in order to express uncertainty) employing various Monte-Carlo sampling algorithms, e.g. rejection sampling, forward sampling, and Metropolis-Hastings.

The MAPL classes and class relationships are all aligned with ArchiMate[4] (an Open Group standard), while the attributes adhere to each of the goals/analyses selected. E.g. for availability the user can enter information about *time between failures*, *time to repair*, and *observed availability* (see Figure 1 for an example). Either one can enter deterministic numbers, e.g. "I know that the time to repair this application component is 10 hours," or distributions, e.g. "I am not sure how often this infrastructure components goes down but it is normally distributed around once a month."



perty	Value
Bins	20
Derived	true
Evidence	
PercentageGraph	false
Statement	self.subModel(self.is...
Type	Real
Values	0.9

```
Xtext OCL for 'null' : null
```

Figure 1. A screenshot from EAAT using MAPL. The example shows a small analysis of business process availability.

[1] www.ics.kth.se/EAAT
[2] http://www.uml.org/
[3] http://www.omg.org/spec/OCL/
[4] http://www.opengroup.org/subjectareas/enterprise/archimate

In addition to MAPL, we have also developed several other P^2AMF languages that can be used with the EAAT. The most extensive one being the Cyber Security Modeling Languages (CySeMoL) [8-9].

References

[1] Pontus Johnson, Robert Lagerström, Per Närman, and Mårten Simonsson, ”Enterprise architecture analysis with extended influence diagrams,” *Information Systems Frontiers*, vol. 9, no. 2-3, pp. 163-180, 2007.

[2] Pontus Johnson and Mathias Ekstedt, Enterprise architecture: models and analyses for information systems decision making, Studentlitteratur, 2007.

[3] Pontus Johnson, Robert Lagerström, Mathias Ekstedt, and Magnus Österlind, *IT Management with Enterprise Architecture*, ePub, http://www.ics.kth.se/MAP.pdf, 2013.

[4] Teodor Sommestad, Mathias Ekstedt, and Hannes Holm, ”The cyber security modeling language - a tool for assessing the vulnerability of enterprise system architectures," *IEEE Systems Journal*, vol. 7, no. 3, pp. 363-373, 2013.

[5] Per Närman, Hannes Holm, Pontus Johnson, Johan König, Moustafa Chenine, and Mathias Ekstedt, ”Data accuracy assessment using enterprise architecture,” *Enterprise Information Systems,* vol. 5, no. 1, pp. 37-58, 2011.

[6] Ulrik Franke, Pontus Johnson, and Johan König. ”An architecture framework for enterprise IT service availability analysis,” *Software & Systems Modeling*, vol. 13, no. 4, pp. 1417-1445, 2014.

[7] Pontus Johnson, Johan Ullberg, Markus Buschle, Ulrik Franke, and Khurram Shahzad, “An architecture modeling framework for probabilistic prediction,” *Information Systems and e-Business Management*, vol. 12, no. 4, pp. 595-622, 2014.

[8] Hannes Holm, Khurram Shahzad, Markus Buschle, and Mathias Ekstedt, “P^2CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language,” *IEEE Transactions on Dependable and Secure Computing*, no. 99, 2014.

[9] Robert Lagerström, Pontus Johnson, and David Höök, “Architecture analysis of enterprise systems modifiability - models, analysis, and validation,” *Journal of Systems and Software* vol. 83, no. 8, pp. 1387-1403, 2010.

Enterprise Architecture Analysis with Production Functions

Ulrik Franke

FOI Swedish Defence Research Agency

ulrik.franke@foi.se

Enterprise Architecture (EA) is a discipline designed to cope with the complexity of modern enterprises at the intersection of technology and business operations. Enterprises are increasingly dependent on information technology (IT) which is being used for sales, production, human resources management etc. Furthermore, enterprise decision-makers are increasingly concerned with the IT/business interface. There is also a trend towards more data-driven decision-making, and studies show that this pays off in terms of productivity. This presentation ties these two strands – the IT/business interface and data-driven decision-making – together by demonstrating how EA models can be enriched with the production function concept from microeconomics, enabling new and relevant kinds of analysis.

In microeconomics, production functions are used to summarize the production possibilities of a firm; expressing technologically feasible combinations of inputs and outputs. Inputs are typically capital, labor, IT etc. whereas outputs can be any goods or services produced. Mathematically, the production of the maximum amount of output with the minimum amount of inputs can then be formulated as a constrained optimization problem. Optimal solutions are characterized by the condition that the economic rate of substitution (ERS) – the rate at which two inputs can be substituted for each other maintaining a constant cost – equals the technical rate of substitution (TRS) – the rate at which those same inputs can be substituted for each other maintaining constant level of output. This is intuitive: whenever the ERS and the TRS are not equal, either the same output can be produced cheaper, or a larger output can be produced at the same cost, implying that the current state is not optimal.

Now, the useful aspect of this is that there is a large body of literature on production economics, e.g. investigating IT productivity, where production function parameters have been estimated empirically. This means that proposed to-be EA models can be analyzed, using the econometrically estimated parameters, to produce a verdict on whether they are optimal in the sense of production economics.

The approach is demonstrated through three thought experiments:

1. The first example concerns extensive or intensive growth strategies for a company. In this case, we consider a financial analysis firm, operating with two production factors: IT and labor. Thinking about growth strategies, two alternative to-be architectures have been prepared; one for extensive and one for intensive growth. The *extensive* alternative involves hiring more financial analysts, and more first-line IT support. The *intensive* alternative involves firing junior financial analysts, whose work can be automated, replacing them with a Business Intelligence cluster, cutting back on IT support, and hiring some more qualified IT staff to manage the cluster. How should the chief information officer (CIO) evaluate these alternatives? The two growth strategies are basically about re-adjusting the balance between the IT and labor production factors. Thus, the two architectures can be evaluated by combining econometrically estimated parameters from the literature with the actual case data from the architectures (i.e. quantities of staff and IT equipment) to find the ERS and the TRS. The closer the TRS/ERS ratio is to unity, the closer the proposed architecture is being optimal.

2. The second example concerns strategies for high availability IT services. Consider a business critical IT service with high requirements on availability. The service might be part of an industrial or business process where downtime entails large costs, or it might be part of some critical infrastructure where downtime poses significant risks to human life. Assuming that we need to meet a certain availability level (e.g. 99.81 %), should a manager aim for a longer mean time to failure (MTTF) or a shorter mean time to recovery (MTTR)? Assuming that capital can buy better hardware (or more of the same, to build redundancy), thus increasing the MTTF, and that labor can be used to monitor the system and take swift action if it fails, thus decreasing the MTTR, this problem can formulated and solved as a constrained production economics optimization problem. Given capital and labor costs, this allows for solving the trade-off between MTTR and MTTF, i.e. reaching a given availability level in the cheapest way, or, alternatively, spending a fixed budget to achieve the highest possible availability.
3. The third example concerns optimal composition of a military unit. A military unit can consist either of a few pieces of high quality equipment manned by a few soldiers, or of many pieces of low quality equipment manned by many soldiers. How can the trade-off between quality and quantity be made? Based on the Lanchester differential equations describing attrition warfare, this problem can be formulated as a constrained production economics optimization problem. Given a reference enemy (red) force (i.e. quantities of soldiers, tanks, infantry fighting vehicles and artillery pieces), and assumptions about relative qualities of blue (our) and red equipment, the relative fighting strengths of blue force architectures (i.e quantities of soldiers and equipment) can be evaluated. If, additionally, assumptions are made about the prices of buying more quality and quantity, optimal compositions can be found.

The common factor in all three examples is that information already present in EA models (e.g. the number of employees, the IT systems used in operations, or the number of tanks in a military unit) can be further exploited using the production function concept from microeconomics. This allows a number of business analysis concerns to be addressed, following the literature on production economics. The approach is demonstrated with three examples.

Apart from the third example above, which is novel, the proposed presentation is based on the following previous publication:

- Ulrik Franke. Enterprise Architecture Analysis with Production Functions. In *IEEE 18th International Enterprise Distributed Object Computing Conference (EDOC 2014)*, pages 52–60. IEEE, 2014. doi:10.1109/EDOC.2014.17.

Enterprise architecture with executable modelling rules: A case study at the Swedish Defence Materiel Administration

Daniel Oskarsson

FOI Swedish Defense Research Agency

daniel.oskarsson@foi.se

An enterprise architecture (EA) model is composed of symbols (boxes, arrows, words, etc.) that combine to make claims about the business being modelled. How the symbols combine to express meaningful statements is given by the model's *semantics*. Part of the semantics is typically governed by an explicit *meta-model*; ideally, the rest is governed by informal or tacit agreement across modellers and model users.

As long as the model semantics is thus fully determined, explicitly or informally, it can, at least in part, be captured by formal (and thus executable) modelling rules, e.g. in OCL, forming an *extended meta-model* which can in turn be used to automatically verify that the model complies with the semantics.

But what if part of the semantics are not even implicit but thoroughly undetermined? In a typical modelling scenario, with multiple modellers and stakeholders modifying and interacting with the model over a span of time, the risk is then that semantic underdetermination leads to model inconsistencies that go undiscovered by automatic compliance checking, since that which is undetermined cannot be formalized into compliance rules.

Fortunately, the very process of formulating executable rules lends itself to weeding out vagueness: Since a rule, to be executable, must be expressed in precise concepts, formulating the rule entails specifying those concepts that are yet not precise.

To secure the participation and involvement of a broad range of stakeholders in the rule formulation process, and thus ultimately to ensure the quality of the extended meta-model thus produced, it helps if the rules, in addition to being formal, are expressed in a language that resembles natural language.

So, resting upon the assumptions that (1) it is useful to continually submit an EA model to validation against semantic rules, (2) the process of expressing formal rules has the effect of forcing semantic specification, bringing value by precluding model vagueness, and (3) natural language rules improve quality by involving a wider range of stakeholders in the formulation of the semantics, we propose a method for automatic model validation and continual semantic specification based on semantic rules expressed in a controlled natural language.

The method has rule authors formulate constraints on the architecture model using a controlled natural language to relate terms given by the meta-model. Specifically, SBVR (Semantics of Business Vocabulary and Rules) Structured English is used as a controlled natural language. An example rule is thus:

“A system that is used by a combat unit at a date must have a use phase that is active at that date.”

Rules such as this one are automatically compiled into SQL queries that generate lists of violations against them from the architecture repository. The compilation rests on a concept model that maps all meta-model terms that are used in the rules (nouns and verbs such as “system” “life-cycle phase is active at date”, etc.) to corresponding data in the repository.

Generating lists of rule violations has its own model validation purpose, but the idea is also that the process of formulating and executing rules itself triggers discussions about the intended meaning of terms that eventually converge upon semantic agreement – yielding a more stable and thought through meta-model.

We develop and formalize our assumptions into a set of hypotheses about the role of modelling rules in EA projects that motivate the proposed method. We then evaluate the hypotheses qualitatively and to some extent quantitatively in the context of two EA projects at the Swedish Defence Materiel Administration (FMV). The case studies lend support to the hypotheses.

The full article is to be published in CAiSE 2015 Workshop Proceedings.

Architectural Concepts for Federated Embedded Systems

Jakob Axelsson and Avenir Kobetski

Swedish Institute of Computer Science (SICS)

E-mail: jakob.axelsson@sics.se, avenir.kobetski@sics.se

Introduction

With the increasing availability of affordable communication services, the possibility to connect different systems to each other has grown in importance, and led to large interest from industry and academia in the challenges of creating *systems-of-systems* (SoS). Some characteristics of an SoS is that a number of independent systems are connected to create emergent functions and properties at the SoS level. Each constituent system has a value on its own, even when used outside the SoS, and may be delivered and deployed independently by different manufacturers [5].

Recently, SoS have also been given attention in the area of *cyber-physical systems* (CPS). Here, the traditional embedded systems (ES), where electronics and software of a product interact with the physical world through sensors and actuators, are extended with connectivity [4]. Due to their interaction with the physical world, CPS are often subject to other, and more stringent, requirements than other software-based systems, including dependability, security, timing requirements, product cost, and various life-cycle related qualities.

In this paper, we will present findings related to a kind of SoS in the CPS area, which we call *federated embedded systems* (FES). In a FES, the creation of the SoS is based on connecting the embedded systems (ES) in each product with each other, and also potentially with software running on servers outside the embedded systems. In this way, it becomes possible to create services on top of a combination of products, using a concept of *plug-in software* which can be dynamically added to the ES of a product. We call such an SoS a federation, since the constituent systems choose to participate voluntarily, for mutual benefit of the participants. The federation services are the intended emergent functions of the SoS. A key benefit of FES is that the adaptation of a particular product to a certain federation should be flexible and dynamic, allowing the addition of services that were not thought of at the time of designing the products, something that is not possible with a pre-defined communication interface.

Research questions

As described in [3], one of the key success factors in developing FES is the software architecture. In fact, there are two architectures that are relevant, namely the base *product architecture* whose ES will be enabled for participating in federations, and the *federation architecture*, that structures the services provided by that federation, which may need to include a large number of different products. The product architecture can be thought of as an infrastructure on which FES are built, whereas the federation architecture is the applications using the infrastructure.

Both these architectures have a number of challenges, some of them shared, and others individual. Therefore, the two research questions of this paper are as follows:

1. What are the important architectural characteristics needed to enable a product for FES?
2. What are the important architectural characteristics of a federation service to be built on products enabled for FES?

To arrive at the characteristics, it was first necessary to identify key stakeholders and their concerns, which can be used as the rationale for architecture decisions. A series of interviews were conducted with industry representatives, leading to the identification of a number of needs and concerns in the areas of business models, architecture, and process, methods and tools [3].

Concerns

The main concerns of different stakeholders on the architecture were elicited as a set of qualities that are essential in FES. Some of the qualities are equally relevant for both the federation and product architectures, and others are primarily relevant for one of them (although there may be minor implications also for the other). There are also tradeoffs between some of the qualities which contradict each other. Since the concerns described here are valid for a general FES, a specific instance would normally have many other concerns that relate to its functions and its environment. Table 1 gives a summary of the concerns, and to which of the two architectures they primarily apply.

Table 1. Summary of architectural concerns for FES.

Acronym	Concern	Product architecture	Federation architecture
D	Dependability	x	x
S	Security	x	x
A	Assurability	x	x
V	Variability	x	x
C	Composability		x
P	Portability	x	x
O	Openness	x	
F	Flexibility	x	
R	Resource usage	x	x
M	Maintainability		x

Architectural concepts

After having captured the architectural concerns, a number of key design decisions were made in the product and federation architectures, with the rationale for those decisions expressed in relation to the concerns. The main decision areas included: programming concepts, to allow efficient FES development, especially plug-in software and component-based software; federation architecture concepts, such as federation operation management functions and federation life-cycle management functions; and product architecture components, with a focus on the external communication manager, the plug-in runtime environment, configuration support, and simulation. These concepts are on the level of a reference architecture, since they deal with the concepts related to FES, whereas all concrete architecture instances would also include many other aspects specific to its functionality and domain. The main constructs in the architectures are shown in Figure 1. In the figure, the shaded parts are belonging to the federation architecture and the rest belong to the product architecture. The letters in black circles indicate concerns, using the acronyms indicated in Table 1.

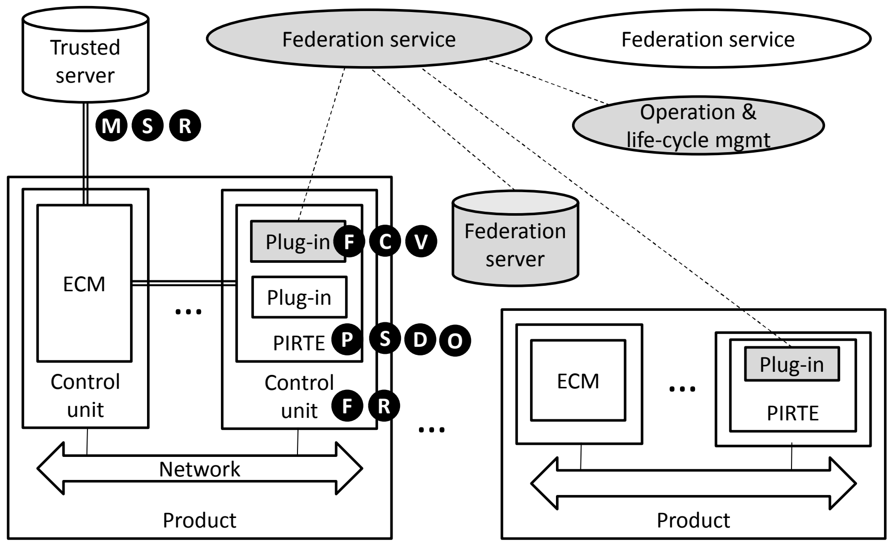


Figure 1. Overview of FES architectures, with relations to concerns.

Validation

To validate the architectures, and provide a means to gather more empirical data on FES development and usage, a demonstrator has been created. It is called the Mobile Open Platform for Experimental Design (MOPED) [2], and consists of a model car in scale 1:10, which is equipped with a distributed computer system consisting of three control units based on Raspberry Pi hardware, and connected via Ethernet. Two of the control units execute software based on the AUTOSAR automotive software standard, and the third is based on Linux, which makes the configuration very representative of the software in a real vehicle. Each AUTOSAR node has various sensors and actuators, whereas the Linux node acts as a telematics unit responsible for external communication. The AUTOSAR nodes contain a runtime environment in the form of a Java sandbox where plug-ins can be installed, and the Linux node contains the external communication manager which allows the installation and management of plug-ins to be controlled from a trusted server.

The full version of this paper is available in [1].

References

[1] Axelsson, J. and Kobetski, A. Architectural Concepts for Federated Embedded Systems. Int. Workshop on Software Engineering for Systems of Systems, 2014.

[3] Axelsson, J. et al. 2014. A Mobile Open Platform for Experimental Design of Cyber-Physical Systems. Euromicro SEAA, 2014.

[4] Axelsson, J. et al.. Characteristics of software ecosystems for Federated Embedded Systems: A case study. Information and Software Technology, Apr. 2014.

[5] Broy, M. and Schmidt, A. Challenges in Engineering Cyber-Physical Systems. Computer. 47(2):70–72, 2014.

[14] Maier, M.W. Architecting principles for systems-of-systems. Systems Engineering. 1, 4 (1998), 267–284 , 1998.

Systems-of-Systems Reliance on Emergent Properties

Patrizio Pelliccione

Gothenburg University

patrizio.pelliccione@cse.gu.se

A System of Systems (SoS) is a collection of often pre-existing and/or independently owned and managed systems that collectively offer a service that emerges from their collaboration [1, 2, 3]. Prominent examples of SoSs include intelligent transport systems, integrated air defense networks, applications in healthcare and emergency response. The units that compose an SoS are systems themselves and are called constituents. SoSs may evolve as triggered by changes in their operating environment and/or in the goals of the autonomous constituent systems [4, 3]. Evolution might affect the structure and composition of the constituents, functionalities offered, and/or the functionalities quality. Collaboration between SoSs enables new capabilities, but interdependency implies that failures can cascade throughout the SoS, creating additional system failures or development delays.

An open problem of SoSs is how to provide justification of the reliance on emergent properties [3]. Emergent properties are the result of synergistic collaboration between constituent systems, and are those properties that cannot be expressed at the level of a single constituent system, but require observations of phenomena at the SoS boundary [3]. Emergence can be either anticipated, meaning that it is defined at design-time, or unanticipated, meaning that it is not purposely or consciously designed-in or surprising to the developers and users of the SoS. Consequences of the unanticipated emergent behavior may be viewed as negative/harmful, positive/beneficial, or neutral/unimportant by stakeholders of the SoS [5]. Existing approaches to design and verify dependable systems work pretty well with closed and unchanging systems. These techniques become inadequate for assessing and justifying SoS reliance on emergent properties since SoSs are composed of autonomous constituents whose behaviour can neither be predicted nor controlled [6]. There is the need of maintaining high quality and supporting evolution during the entire life cycle [7]. For this reason we will measure the SoS reliance on emergent properties in terms of SoS resilience, which is the ability to deliver, maintain, improve services when facing threats and evolutionary changes [8]. Investigation is needed to provide ways for characterizing, quantifying and measuring a system behavior in the presence of perturbations [6].

The overall goal of our proposal is to provide methods and tools for assessing and justifying SoS reliance on emergent properties. The strategy that we will follow is to provide three combined contributions: (i) characterizing emergent properties and helping SoS engineers to easily and correctly specify those properties; (ii) definition of a framework to provide theoretical foundations for the evolution of the SoS; (iii) runtime verification methods to check that an actual execution of the SoS performs satisfactorily. SoS reliance on emergent properties will then be measured in terms of resilience and through metrics, which will make use of the framework and of runtime verification techniques.

The framework will make possible the understanding of the effects of both foreseeable and unforeseeable changes in the SoS operating environment, as well as in the goals of the owners of the autonomous constituent systems. Runtime verification methods will exploit the framework to better interpret emerging behaviours and will check whether the actual execution is invalidating desired properties and contracts.

References

[1] M. W. Maier. Architecting principles for systems-of-systems. Systems Engineering, 1(4):267–284, 1998.

[2] M. Janishidi. System of Systems - Innovations for 21st Century. In Industrial and Information Systems, 2008. ICIIS 2008. IEEE Region 10 and the Third international Conference on, pages 6–7, Dec 2008.

[3] J. Fitzgerald, P. Larsen, and J. Woodcock. Foundations for Model-Based Engineering of Systems of Systems. In Complex Systems Design & Management, pages 1–19. Springer International Publishing, 2014.

[4] J. Boardman and B. Sauser. System of Systems - the meaning of of. In System of Systems Engineering, 2006 IEEE/SMC International Conference on, April 2006.

[5] Department of Defense, United States of America. Directions in Systems of Systems Engineering - Report from the Work. on Syn. among Proj. and Direct. In Advan. Syst. Eng., http://cordis.europa.eu/fp7/ict/embedded-systemsengineering/home_en.html.

[6] L. Strigini. Fault Tolerance and Resilience: Meanings, Measures and Assessment. In Resilience Assessment and Evaluation of Computing Systems, pages 3–24. 2012.

[7] U. Goltz, R. Reussner, M. Goedicke, W. Hasselbring, L. Märtin, and B. Vogel-Heuser. Design for future: managed software evolution. Computer Science - Research and Development, pages 1–11, 2014.

[8] ReSIST (Resilience for Survivability in IST), Selected current practices. Deliverable D39, European Network of Excellence 2009.

The notion of ‘Systems of Systems’ should be abandoned

Darek M. HAFTOR

Linnaeus University

darek.haftor@hotmail.com

This talk aims to provide offer our key argument for why the notion of a “*system of systems*” is inadequate for its aim and should be replaced with a more adequate conception. That message is based both on our key theoretical insights into the notion of ‘*System of Systems*’ and our empirically experiences that started in the 1990’s, with the doctrine change of the Swedish Defence, and continued with the emergence of digital businesses from the late 1990’s to the present day.

Both these and many other contexts, for example airlines and transportation, healthcare and the financial institutions, military operations and various facets of digital business practices, manifest an emergence of increased *interconnectness* and *changeability* of physical, virtual and organisational entities, giving rise to an *aggregated behaviour* of these entities which is perceived as unpredicted and sometimes undesirable, such as the recent financial crisis. In order to understand and possibly influence the behaviour of a dynamic aggregate, constituted by interacting entities, the notion of ‘*System of Systems*’ (SoS) was proposed in the 1990’s (e.g. Maier, 1998 ; Sage & Cuppan, 2001; DeLaurentis, 2005; Jamshidi, 2008; Luzeaux & Ruault, 2010) and has today its own journal, the ‘*International Journal of System of Systems Engineering*’. A chief motive stems from the perceived success of the notion of the ‘*system*’ as such, and the various practical systems tools that have been advanced since the Second World War.

While the notion of a ‘*system*’ as such, regarded as a whole manifesting characteristics that cannot be observed in its parts, originates as early as in the writings of Aristotle, it is its re-emergence in the 20th century that has led to various methodological advancements. L. von Bertalanffy (1901-1972) often considered the father of ‘General Systems Theory’ (von Bertalanffy, 1968, 1972; Hammond, 2003) – which offers conceptions that are not very general or theoretical – re-introduced the old Aristotelian notion that a ‘*whole is more than the sum of its parts*’. While von Bertalanffy was motivated by problems in biology (ibid.), i.e. how to understand the behaviour of a living organism, the chief proposition of the notion of a system (i.e. emergent characteristics of a whole) was re-interpreted (e.g. Simon, 1962; Morin, 1977; Le Moigne, 1990; Klir, 1991; Holland 2006) and absorbed by various other disciplines (Hamond, 2003), both in basic sciences – i.e. physics, chemistry, biology, psychology, sociology and economics – and particularly in the so-called applied sciences – i.e. operations research (e.g. Ackoff et al., 1957), systems analysis (e.g. Checkland, 1978; Miser & Quade, 1985), systems engineering (e.g. Hall, 1962,), and various domains of social analysis and planning (e.g. Forrester, 196; Churchman, 1971, 1979, Ackoff, 1981, Checkland 1981; Ulrich, 1983, 1987) and more recently management of economic organizations (e.g. Milgrom & Roberts, 1995; Porter & Siggelkow, 2008). The success of the *systems conception* applied to human affairs – whether they be military, industrial and business, or public – has been manifested many times in practice: e.g. the establishment of the ‘airlift’, after the Berlin Wall was erected by the Soviet powers in 1961, which at its peak required a start and landing of a supply aircraft at the West Berlin airport every 90 seconds; or the construction and despatch of space shuttles to the moon, along with many other interventions.

It is then not so odd that the proposed notion of a ‘*system of systems*’ has attracted the attention and interest of those concerned with domains where *numerous* man-made entities – be they physical, virtual, organisational – interact and generate a joint behaviour that challenges our understanding and influence (e.g. Jamshidi, 2008); for example the expansion of health care from the provision of pure medical diagnoses and treatment of an individual conducted by a single physician to situations that include various dietary, mental, social, physical and economic facilities and their specialist actors, all formed in a temporary and dynamic network with the common end of helping an individual. The key message of systems thinking – i.e. that the interaction of parts (components) generates a global behaviour that cannot be derived from any of its parts alone – is often regarded as highly pertinent also for those contexts where numerous entities (for example military aircraft and military submarines), initially regarded as having their own identities interact in a manner to produce an outcome that none of them could achieve on its own.

However, the notion of a *system*, being based on the biotic root-metaphor of an *organism* (e.g. von Bertalanffy 1968, 1972; Checkland 1981; Hamond, 2003), assumes that a system's parts are all perfectly aligned with its overall purpose, and therefore a system's constituting parts are regarded to be fully subordinated to the whole of which it is a part – as is the case with organs in the human body: the heart, lungs or kidneys make sense only in the context of the human body and lack thus any meaningful identity that is independent of its whole, i.e. when taken outside of the human body. Now, the very specific contribution and peculiarity of the notion of a system also makes the notion of a 'system of systems' an antinomy, or a contradiction, in the sense of the expression "*a married bachelor*". This is so as a system, per definition, cannot consist of other systems, it can only be constituted by system *parts* that unlike the systems as such, lack their own independent identity. This ontological antinomy is unfortunately disregarded in the current *System of Systems* discourse (e.g. Jamshidi, 2008), which we regard as a dangerous tendency, as a key assumption of the notion of a *System of Systems* may generate conceptions and perceptions that fail the very need that originated the use of that notion, namely to perceive and conceive complexities that emerge from a number of entities posing their own identities, and interacting with each other thereby producing emergent global characteristics. A tragic example of such a failure is the recent disaster of the Germanwings Flight 9525, bound from Barcelona to Dusseldorf, where the co-pilot deliberately flew the Airbus A 320 into a mountainside in the French Alps causing the death of 150 people – clearly, assuming that the co-pilot of an aircraft lacks his or her own identity, being part of the larger system, deviates from the actual empirical experience and thereby exercises a dangerous reductionism.

Our position is that the notion of a *System of Systems* should be dismissed as such, and other notions should instead be advanced to address the empirical challenge at hand. To that end we are in the process of exploring two such alternative notions (Haftor & Kurti, 2014): the notions of an *assemblage* (DeLanda, 2006) and the notion of *encaptic relations* (Dooyeweerd, 1997). To exemplify the latter, a small rock in a bird's gizzard may assume a function in the bird's digestive process. The rock is not a part of the bird, rather it assumes a passive function and the rock can exist without the bird yet it cannot perform the same digestive function without the bird. In such whole-whole relation, one whole is governed or obeys one kind of norms or laws while the other whole is governed or obeys another kind of norms or laws; this means that there is a significant difference in the nature of the two entities and therefore these should be conceived in terms of *encaptic relations*.

The ontological antinomy inherent in the notion of a 'System of Systems' also generates epistemological and ethical concerns. Epistemologically speaking, the experience of whole-whole relations suggests that novel processes of conception are required in order generate knowledge thereof. The modernist ambition of perfect and valid knowledge seems untenable, rather we have to accept the aspiration of feasible knowledge (Simon, 1957; Churchman 1971; Le Moigne 1994, 1995) together with critical reflections upon its limitations (Churchman, 1971; Ulrich, 1983, 1987). Morally speaking, the key question is who is responsible in and for situations where numerous whole-and-whole relations, between both human and non-human entities, generate emergent behaviour that may have undesirable and indeed harmful consequences (Floridi, 2013)?

Clearly, more important research needs to be done to produce conceptions that may support our understanding of situations where numerous wholes interact with each other and generate emergent characteristics. The key message here is that the current notion of ‘System of Systems’ should be abandoned due to its inherent antinomy, and alternative conceptions should be advanced to overcome its limitations.

References

Ackoff, R.L., C. W. Churchman and E. L. Arnoff (1957). *Introduction to Operations Research*. Wiley, New York.

Ackoff,R.L.(1981).*Creating the Corporate Future*. Wiley, New York.

Bertalanffy, Ludwig von (1968). *General system theory: foundations, development, applications*. Braziller, New York.

Bertalanffy, Ludwig von (1972). The History and Status of General Systems Theory. The Academy of Management Journal, Vol. 15, No. 4, 407-426.

Checkland, P.B. (1978). The origins and nature of “hard” systems thinking. Journal of Applied Systems Analysis, 5, 2.

Checkland,P.B.(1981).Systems Thinking, Systems Practice. Wiley, New York.

Churchman, C.W. (1971). The Design of Inquiring Systems, Basic Books, New York.

Churchman, C.W. (1979). The Systems Approach and Its Enemies. Basic Books, New York.

DeLanda, M. (2006). A New Philosophy of Society: Assemblage Theory and Social Complexity, London & New York: Continuum.

DeLaurentis, D. (2005). Understanding Transportation as a System of Systems Design Problem. 43rd AIAA Aerospace Sciences Meeting, Reno, Nevada, January 10–13. AIAA-2005-0123.

Dooyeweerd, H. (1997) [1955]. A New Critique of Theoretical Thought, Edwin Mellen. Volume III: The Structure of Individuality of Temporal Reality.

Floridi, L. (2013). Distributed Morality in an Information Society. Science and Engineering Ethics, 2013, 19.3, 727-743.

Forrester, J.W.(1961). Industrial Dynamics. Wright-AllenPress, Cambridge.

Haftor D.M., Kurti, E., (2014). Toward Post Systems Thinking in the Conception of Whole-Part Relations. In: Rathbone, M., von Schéele, F., Strijbos, S. eds. Proceedings of the 19th Annual Working Conference of the International Institute for Developmental Ethics, 6-9 May, Maarssen. Rozenberg.

Hall, A.D. (1962). A Methodology for Systems Engineering. D. van Nostrand Co., Princeton, NJ.

Hammond, D. (2003). The science of synthesis. Exploring the social implications of general systems theory. University Press of Colorado, Colorado

Holland, J. H. (2006). Studying Complex Adaptive Systems. *Journal of Systems Science and Complexity* 19 (1): 1-8.

Jamshidi, M. ed., (2008). *System of Systems Engineering: Innovations for the Twenty-First Century*. Wiley

Klir, J. (1991). Facets of Systems Science. Plenum, New York

Le Moigne, J.L. (1990). La modélisation des systèmes complexes, Éd. Dunod, Paris.

Le Moigne, J.L.(1994). Le constructivisme, Tome 1: Des fondements, ESF éditeur, Paris.

Le Moigne, J.L.(1995). Que sais - je? Les épistémologies constructivistes, PUF, Paris.

Luzeaux, D., Ruault, J.R. (2010). Systems of Systems. Wiley, New York.

Maier, M.W. (1998). Architecting Principles for System of Systems. *Systems Engineering* 1 (4): 267–284.

Milgrom, P., & Roberts, J. (1995). Complementarities of fit: Strategy, structure, and organizational change. Journal of Accounting and Economics, 19, 179–208.

Miser,H.J.,Quade,E.S.(1985),(eds.). Handbook of Systems Analysis: Craft Issues and Procedural Choices. Wiley, New York.

Morin, E. (1977). La Methode. 1. La Nature de la nature. Seuil, Paris

Porter, M., & Siggelkow, N. (2008). Contextuality within activity systems and sustainability of competitive advantage. The Academy of Management Perspectives, 22(2), 34-56.

Sage, A.P., C.D. Cuppan. (2001). On the Systems Engineering and Management of Systems of Systems and Federations of Systems. *Information, Knowledge, Systems Management*, Vol. 2, No. 4, 2001, pp. 325-345.

Simon, Herbert (1957). A Behavioral Model of Rational Choice, in Models of Man, Social and Rational: Mathematical Essays on Rational Human Behavior in a Social Setting. New York: Wiley.

Simon, H.A. (1962). The Architecture of Complexity. Proceedings of the American Philosophical Society, Vol. 106, No. 6. (Dec. 12, 1962), pp.467-482.

Ulrich,W. (1983). Critical Heuristics of Social Planning: A New Approach to Practical Philosophy. Haupt, Bern.

Ulrich,W. (1987). Critical Heuristics of Social System Design. European Journal of Operation Research.31:276.

The role of Human Factors in Systems Engineering, mutual interest and effective co-existence

Lars-Ola Bligård and Robert Nilsson

Chalmers University of Technology, Volvo Car Group

lars-ola.bligard@chalmers.se, Robert.nilsson.2@volvocars.com

Introduction

This abstract will explore the role of Human Factors in Systems Engineering based on the PU^{2}B-modell, Product Utility Usability and Business Model (Bligård, Nilsson, 2015). This is of interest since the field of Systems Engineering and Human Factors has a mutual interest to influence future Product Development. By this clarification the authors state that the same relationship most likely will be applicable for Systems of systems. The last decades' technological advances that enabled smaller electrical components and introduced software as a common part of products has resulted in products are both inter and intra connected to a higher degree. This complexity has led to increased need of understanding product perspectives and structure during development. Some of these issues are not new for the Human Factors domain and established theories and methods could therefore be of interest to highlight in Product Development. The field of Human Factors and Systems Engineering are in some sense closely related. They share several theoretical perspectives e.g. Systems Theory. In the same time has the structured approach of Systems Engineering, which is providing an explicit structure for life cycle perspective put Human Factors in a natural context for product development. It is the authors' strong belief that there is an opportunity to combine the two fields to get the most of both. How they can relate to each other is therefore of most interest, especially if it further enables clarifying perspectives on Systems of systems.

Theoretical background

The assumption of mutual interest of Human Factors and Systems Engineering to co-exist efficient is the common objective to affect system design. Human Factors is defined as "the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimize human well-being and overall system performance" by the International Ergonomics Association.

Sprung from basic elements of Model Based Engineering the PU^{2}B-modell was developed. The main objective with developing the model was to allow quick and easy access for roles of; business analyst, system responsible, project leaders, system architects and not the least human factors engineers. The authors have experienced that although there are several theories and methods that guide product development there is no method or structure that connects the areas of Human Factors and Systems Engineering in this explicit and useful way.

The PU^{2}B-modell presents how Human Factors and the structured approach of Systems Engineering can be combined to achieve higher potentials in Product Development. The model connects Human Factors, representing the customer and user perspective, to a product development technical perspective related to system optimization. The main contribution from the field of human Factors is the application of theories related to Activity Theory. Activity theory explains the relationship of activity that implies a goal for the function that can be used and the object as an artefact to mediate use. Another major contribution was the recognition that the same kind of design issues has to be solved in different abstraction levels, seen from a structural perspective of the system.

Description of the PU^{2}B-model

The PU^{2}B-model connects Human Factors, representing the customer and user perspective, to a product development technical perspective which is related to system optimization. The three abstraction levels covered by the PU^{2}B-model are: System definition, Product definition and Architecture definition. System definition explains the system boundaries and expected outcome of the system. Product definition clarifies what's expected by the product or concept in relation to the system. Architecture definition defines the top layer of the solution that constitutes the product and is to be developed or implemented.

Each of the abstraction levels have model objects that are of the types; structure objects, function objects and activity objects. Examples of structure objects are; System stakeholders (System definition), System objects (System definition), Product domain objects (Product definition), States and modes (Product definition) and Logical architecture (Architecture definition). Depending of the abstraction level the object types will contain different kind of information and be linked to other objects to enable a top down design perspective. Although the PU^{2}B-model was developed for top-down perspective it can be linked backwards enabling "bottom-up" perspective.

Discussion

The main objective with developing the model was to allow quick and easy access for roles of; business analyst, system responsible, project leaders, system architects and not the least human factors engineers, by providing structure and base for reasoning of functionality and transforming business needs to product construction views.

PU^{2}B-model is a god tool for managing customer value and the value-creating activities of the user. The relationship between PU^{2}B-model and Scrum is that PU^{2}B-model sets the framework for an Agile approach. A project can turn into an Agile approach when the system architecture is mainly set. The authors argue that the PU^{2}B-model mechanisms, based on Activity theory, can enable structure for systems of systems to communicate interest and goals.

References

Bligård, L.-O., Nilsson, R., (2015), PU2B-modellen - En introduktion till Model Based Systems Engineering (MBSE) utifrån användarcentrerad systemdesign, Göteborg, Chalmers tekniska högskola.

From Conventional to Adaptable Manufacturing Paradigms – A Systems Perspective (An Industrial Case Study)

Afifa Rahatulain and Harold ‘Bud’ Lawson

SenseAir AB, Lawson Konsult AB

afifa.rahatulain@senseair.se, bud@lawson.se

This paper presents a case study of the application of *systems engineering* & *systems thinking* tools and methods on a gas-sensors manufacturing company, SenseAir AB. Like other Small and Medium Enterprises (SMEs) the company faces certain challenges due to the increased competition in the global manufacturing market. These challenges include but are not limited to; having shorter lead times, reduced down-times and quick adaption to diverse customer demands while striving towards sustainable and low-cost production. However, to embrace the emerging technological innovations in the traditional production setup, identification of the existing strengths and weaknesses and impact analysis of modifications on the system elements is required. Thus, a need to analyze the existing production systems with a holistic view i.e. as a *System of Systems* perspective arises.

SenseAir AB located in Sweden is one of the world’s leading companies in Non-Dispersive Infrared (NDIR) gas sensing technology. The state-of-the-art gas sensors are extensively used in applications for building ventilation, safety, air quality control, automotive industry, mining, agriculture, etc. The emerging market trends and technological innovations have made it inevitable for the company to adapt to advanced manufacturing techniques and enhance its personnel capabilities within production systems and automation [1]. The main emphasis is towards adopting sustainable production solutions considering social, ecological and economical aspects.

The main objective of this case study is to apply the concepts from the systems engineering & systems thinking domains on the production system of SenseAir AB, and to evaluate the possible challenges and opportunities related with a paradigm shift from conventional to advanced, intelligent and adaptable production solutions.

The systems perspective utilized in this case study involves both systems engineering and systems thinking. The concepts that are utilized are derived from the ISO/IEC/IEEE 15288 standard [2] that is concerned with processes for life cycle management. The results though derived from a gas-sensors manufacturing plant, can also be applied for a general analysis of other small and medium sized production systems.

Several modeling tools and methods have been developed over time to assist in adopting the systems approach [3]. In this case study, the following methodology has been adopted for the application of systems thinking & systems engineering theory:

1) Identification of system of interest (SOI) and system boundaries.
2) Recursive decomposition of the system into system elements or subsystems.
3) Arranging in network & hierarchical topologies for understanding structural properties.
4) Development of mental models (system descriptions) using tools such as system archetypes, influence diagrams, rich pictures, Systemigram, root-cause method, system coupling diagrams, link-loops and delays, etc.
5) Mathematical modeling or quantitative analysis.

The application of Systems Engineering & Systems Thinking theory and its models on the production system enabled in identifying the potential risks and hazards associated with the system. The identification of the bottlenecks and the root causes for various production issues also led to the integration of the product design process earlier into the production system using methods such as DFA (Design For Assembly), etc. Potential improvement projects for the product design considering the whole system over the entire life cycle stages were also initiated internal to the organization. The recursive decomposition of the system helped in determining the hidden complexities of the system and initiated some new ideas to tackle these complexities as well as to explore new methods for improving the overall production yield.

Also, the system coupling diagram proved to be a very useful tool for analyzing & coping with different emerging situations utilizing the available system assets. In adaptable systems, since the adaptability is mainly due to the inherent capability of exploiting emergent behaviors in a system [4], this tool could be of particular importance when evaluating a major system change.

The Systemigram provided an overview of the important elements of the system and their relationships. The quantitative analysis performed in this paper on traditional production systems using Simulink/SimEvents can be combined with the SimEvents analysis on intelligent manufacturing paradigms [5] to provide a comprehensive and holistic view of the system.

The comparison of the existing production system life cycle with an adaptable approach (Evolvable Production System – EPS, in this case) also exposed certain opportunities and risks associated with the paradigm shift. In contrast to the current systems, with a linear transition over the life cycle stages, an EPS has a closed circular loop in its utilization stage referred to as the “Evolution” stage [6], as shown in fig. 1. This is a major difference in moving towards evolvable systems from traditional systems. It will not only have an impact on the business model associated with the system but will also contribute significantly towards a more sustainable system by re-utilization of the system modules within the closed loop reducing carbon footprints and raw material usage.

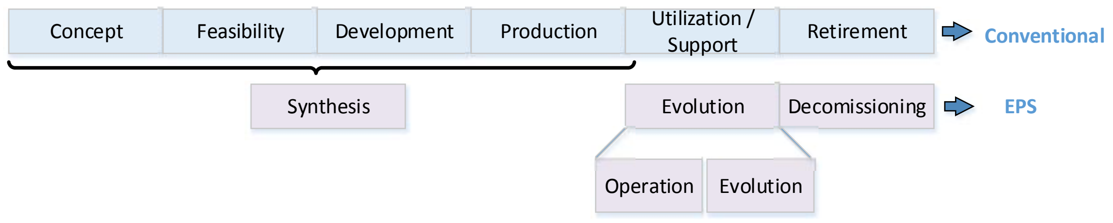


Figure 1. A comparison of the life cycle stages between conventional and adaptable production systems.

The systemic analysis of the system also helped in identifying some of the major challenges associated with the adoption of emerging production paradigms. A few of those challenges are listed as follows:

- Multi-disciplinary information management
- Verification & Validation (V&V) requirements and design tool support
- Modification of existing and development of new industrial standards and protocols
- Business Model Innovation
- System integration, IPR and legislative issues.

The results from this case study shall be further examined to explore the above mentioned challenges. The findings shall also be strengthened using several other test cases and industrial applications. Moreover, the results shall also serve as a basis for developing comprehensive reference architecture for the adoption of the emerging manufacturing paradigms. A change management model and a life cycle management model considering the stakeholders' requirements shall also be developed.

A major part of this work (Title: "Towards Life Cycle Management of Industrial Manufacturing Systems – A Systems Perspective") has been presented at the 9th Annual IEEE International Systems Conference, SysCon'2015, 13-15th April, 2015, Vancouver, British Columbia, Canada.

References

1) Vinnova "UDI Project Information", http://www.vinnova.se/sv/Resultat/Projekt/ Effekta/2011-01544/Hallbara-produktionsformer-for-hogteknologisktillverkning-av-MEMS-baserade-sensorsystem-i-Sverige/, Last accessed March 2014.
2) ISO/IEC/IEEE 15288 (2008): Systems and Software Engineering – Systems Life Cycle Processes, International Standardization Organization / International Electromechanical Commission, 1, rue de Varembe, Geneva 20, Switzerland Std.
3) P. Checkland, "*Systems Thinking, Systems Practice*", John Wiley & Sons Ltd.
4) M. Onori, J. Barata, and R. Frei, "*Evolvable Assembly Systems: Basic Principles*"in Information Technology for Balanced Manufacturing Systems, IFIP, International Federation for Information Processing, Springer US, 2006, vol.220, pp. 317-328.
5) A. Rahatulain, T.N. Qureshi, M. Onori, "*Modeling & Simulation of Evolvable Production Systems using Simulink /SimEvents*", in the 40th Annual Conference of the IEEE Industrial Electronics Society, 2014.
6) B. Lindberg, M. Onori, D. T. Semere, "*Evolvable Production Systems – A Position Paper*", Swedish Production Symposium 2007, SPS'07.

Towards Standard-Based Healthcare Ecosystems of Systems

Konstantinos Manikas

University of Copenhagen

kmanikas@di.ku.dk

Introduction

Software systems have been historically evolving to become more complex and dependent. Nowadays it is rather rare that a system is completely developed from the beginning without reuse of components or parts of other systems. This tendency is also reflected in the popularity of the notion of Systems of Systems (SoS) where the collection of software systems results in new and bigger systems of more complexity that simply the collected systems. Systems of systems typically comply with strict requirements such as security, reliability, and scalability while they are known to be applied to, among other, domains of mission-critical nature [1]. In this position paper, we examine systems of systems in the medical and healthcare domain.

Medical systems are typically characterized by high requirements in security and privacy. These requirements, possibly in a lower grade, are extended to systems surrounding medical systems like apps that inform patients with advanced pacemakers how to detect and improve condition worsening [2]. The need for regulating healthcare systems is eminent [3].

Furthermore, healthcare systems have been growing more interdependent and interoperable with time. This has become more obvious with the increased requirements of centralizing systems as recently demonstrated in the case of the electronic medical records and more specific the example of the Danish healthcare [4]. These requirements of interdependency and interoperability bring healthcare systems closer to software ecosystems, i.e. systems that are characterized by the symbiosis of software and their actors (e.g. companies, organizations) on top of a common technology [5].

Software ecosystems have been traditionally found in non mission-critical domains such as smart phones with the example of the Apple AppStore and software development with the Eclipse platform. However, the notion of software ecosystems is lately becoming popular also in mission-critical domains like the automotive industry [6] and healthcare [7]. Software ecosystems typically evolve around a technological platform that facilitates development and supports the activities of the ecosystem. Recently it has been suggested that software ecosystems can also evolve around a set of standards that take the role of a common platform [8, 9].

Healthcare Ecosystems of Systems

In this paper, we argue for the combination of systems of systems and software ecosystem theories (as already discussed in previous work [10]). Our approach is studying the establishment and evolution of a system of systems in the healthcare domain, i.e. the extended medical domain to include systems that are not medical systems per se but are directly related to medical systems. We identify that an ecosystem of this nature is consisted of software systems that tend to be highly specialized and characterized by high requirements in safety, availability, and privacy. Moreover, these systems are created by several and many times specialized actors [11], thus the network of actors involved can be characterized by several actors with close to equal influence on the ecosystem, or rather by the lack of a dominating actor.

Furthermore, the business models that serve these actors, at least in the case of several European countries, are different than possible traditional software development models where the users pay the cost of the systems. In these ecosystems the involved actors gain revenues for their activity by payments from the state (or other healthcare organizations like insurances) [12]. It is, in other words, a state-funded software ecosystem.

Our approach is that the evolution of systems of this kind around a set of standards would support the activity and foster the well functioning of the ecosystem. We argue that a standard-based ecosystem of systems in the healthcare domain would support better the ecosystem characteristics compared to a traditional ecosystem based on a software platform. A standard-based ecosystem [13]:

- Supports the (co)existence of multiple actors. Especially in the case of the actors having different interests and potentially equal influence to the ecosystem.
- Allows for a wider control and influence of the architectural qualities of the produced systems.
- Facilitates ecosystem orchestration. Taken that the standards of the ecosystem are in place, orchestrations is more oriented towards ensuring standard compliance, thus relieving part of the effort that would normally be required in a traditional ecosystem.

Conclusion

In this paper, we discuss an approach to healthcare systems. Taken that healthcare systems evolve more complex, interdependent, and interoperable with time, we propose the combination of systems of systems and software ecosystems theory to support this evolution. Our approach includes the establishment of a set of standards as a central point that would facilitate system development and actor symbiosis. The establishment of standard-based ecosystem of systems in the domain of healthcare would support coexistence of actors, allow for requirement control, and facilitate the ecosystem orchestration.

References

1. Kotov, V.: Systems of systems as communicating structures. Object-oriented technology and computing systems re-engineering (1997) 141-154 ?
2. Version2: Dingeling: Det er din pacemaker, der ringer. http://www.version2.dk/artikel/dingeling-det-er-din-pacemaker-der-ringer-59022, accessed April 29 (2015) ?
3. Manikas, K., Hansen, K.M., Kyng, M.: Governance mechanisms for healthcare apps. In: Proceedings of the 2014 European Conference on Software Architecture Workshops. ECSAW '14, New York, NY, USA, ACM (2014) 10:1-10:6 ?
4. Kierkegaard, P.: ehealth in denmark: A case study. Journal of medical systems 37(6) (2013) 1-10 ?
5. Manikas, K., Hansen, K.M.: Software ecosystems - A systematic literature review. Journal of Systems and Software 86(5) (2013) 1294 - 1306 ?
6. Eklund, U., Bosch, J.: Architecture for embedded open software ecosystems. Journal of Systems and Software 92(0) (2014) 128 - 142 ?
7. Manikas, K.: Analyzing, Modelling, and Designing Software Ecosystems - Towards the Danish Telemedicine Software Ecosystem. PhD thesis, Department of Computer Science, University of Copenhagen, Denmark (2015) ?
8. Knodel, J., Manikas, K.: Towards a typification of software ecosystems. In: Proceedings of the 6th International Conference on Software Business, Braga, Portugal, June 11, 2015. (2015) ?
9. Jansen, S., Cusumano, M.: Defining software ecosystems: A survey of software platforms and business network governance. In Jansen, S., Bosch, J., Alves, C., eds.: Proceedings of the Forth

International Workshop on Software Ecosystems, Cambridge, MA, USA, June 18th, 2012. Volume 879., CEUR-WS.org (2012) 40 - 58 ?

10. Papatheocharous, E., Axelsson, J., Andrersson, J.: Issues and challenges in ecosystems for federated embedded systems. In: Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems. SESoS '13, New York, NY, USA, ACM (2013) 21-24 ?
11. Manikas, K., Hansen, K.M.: Characterizing the Danish telemedicine ecosystem: Making sense of actor relationships. In: Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems. MEDES '13 (2013) 211-218 ?
12. Christensen, H.B., Hansen, K.M., Kyng, M., Manikas, K.: Analysis and design of software ecosystem architectures - towards the 4s telemedicine ecosystem. Information and Software Technology 56(11) (2014) 1476 - 1492 ?
13. Knodel, J., Manikas, K.: Reference Architectures for Standard-Based Software Ecosystems. Under submission.

Modeling and Simulation of Evolvable Production Systems using Simulink / SimEvents

Afifa Rahatulain, Dr. Tahir Naseer Qureshi, and Prof. Mauro Onori

SenseAir AB, HIAB AB, KTH

afifa.rahatulain@senseair.se, tahir.qureshi@hiab.com, onori@kth.se

High automation costs, shorter lead times, quick adaption to market fluctuations and sustainability are a few of the major challenges faced by the current manufacturing industry to cope up with the variance in customer demands, both in terms of product diversity and quantity. Evolvable Production Systems (EPS) is one of the several emerging approaches targeting these challenges. It provides features such as self-organization and adaptability at the shop floor level through its modular, intelligent & multi-agent based control. In contrast to the existing production systems with limited re-configuration capabilities, an EPS provides the concept of *Plug & Produce* through its distributed architecture allowing dynamic modifications, i.e. addition/removal of system modules in real-time [1].

A few of the latest developments related to EPS include; the concept of a reference architecture [2], an ontology to support evolvable assembly systems [3], utilization of JADE (Java Agent Development Environment) platform for coding and implementation of agents [4], a visualization tool for retrieval of the information exchanged between different agents [5], a methodology for configuring skills of an EPS [6], etc. Although the above mentioned efforts are considerable enough, further work is still required for a wide scale industrial acceptance of EPS to ensure characteristics such as reliability, safety and integrity of the system.

One of the major directions for further work in EPS is towards a well-defined methodology i.e. ``tools and rules'' [7], covering various aspects of the development life cycle ranging from requirements specification and analysis, system and component analysis to testing. For example, it can be useful to analyze the behavior of the overall system which not only includes agents but also characteristics such as robot dynamics modeled using laws of physics. Several tools such as [8] and [9] exist for simulating manufacturing systems with agent-based behavior and their response to changing environments. Due to their scope targeted towards agent-based simulation, additional tools are required for modeling other aspects of the system like physical constraints. To reduce the number of tools, integrated approaches like [10] where Simulink is integrated with an agent based tool can be considered. The integrated approach can on one hand be non-trivial for some tools due to several reasons such as IP (Intellectual Property) issues and on the other hand gives rise to an increased cost. Also any discrepancies in the integration can lead to possible information inconsistency.

This paper explores the possibility of using Simulink as an all-in-one tool for control system analysis of EPS. The main contributions are as follows:

1) Evaluation of SimEvents (a Simulink blockset for discrete events simulation) for modeling and simulation of the agent-based behavior of EPS. The selection of SimEvents is motivated by extensive industrial usage and the support provided by Simulink for different development aspects such as requirement specifications, verification, validation as well as code generation for different processing platforms. Moreover, the integration of system dynamics (e.g. robot kinematics) and other physical constraints with a discrete event system model is also a possibility.

2) Providing guidelines for modeling a generic EPS using SimEvents which can be used to model and analyze a wide range of physical scenarios. These guidelines in future, can serve as a basis for a methodology incorporating automated model generation and a platform for verification and validation of different algorithms related to EPS.

The work in this paper is motivated by [11], [12] and [13] focusing on aspects similar to EPS from other engineering domains such as *scheduling* for air traffic, *self-reconfiguration* in automobiles and generic *resource allocation* from a pool of resources, respectively. The work is also validated and demonstrated by a preliminary case study.

A major benefit of using SimEvents 'Attribute Function' block for different agents is the possibility of testing different algorithms. At the same time the Attribute Function block can be converted into a Simulink's *embedded Matlab function* followed by automatic generation of C/C++ support provided by Simulink. The generated code can later be used with a wrapper function in a Java based environment such as JADE.

Although SimEvents does not have the same level of flexibility as Java based multi-threaded environment it is still usable for the objective purpose i.e. combined simulation of physical dynamics and agent-based behavior. This is especially true for small to medium scale assembly systems with a limited number of machines and hence skills.

Two major limitations in SimEvents are increased simulation time and modeling effort with increasing number of agents and algorithm complexity. While the former is dependent on the processing power of the machine used for simulation, the latter can be eliminated by automatic generation of Simulink model (mdl-file) or a Matlab script (m-file) to generate Simulink models from a GUI such as [5] for multi-agent platforms. The generation of SimEvents model is non-trivial given the fact that Matlab does not provide a well-defined interface / API for SimEvents as it does for its other blocksets. However, the situation might change in future with the increasing demand of SimEvents API.

Some efforts have been carried out in other engineering domains in terms of mapping formal models such as UML to SimEvents [14] as well as automated generation of Simulink models from UML [15], etc. The results from these efforts can be re-utilized to realize system specifications described using formal methods such as UML, if required.

This paper has been presented at the 40th Annual Conference of the IEEE Industrial Electronics Society, IECON'14, Dallas, Texas, USA, October 29th – November 1st, 2014.

References

1) M. Onori, D. Semere and B. Lindberg, " Evolvable Systems: An Approach to Self- X Production", in CIRP- sponsored International Conference on Digital Enterprise Technology, Advances in Intelligent and Soft Computing, vol. 66, 2010, pp. 789-802.
2) M. Onori and J. Barata, " Mechatronic Production Equipment with Process- based Distributed Control", in 9th IFAC Symposium on Robot Control, 2009, pp. 80-85.
3) L. Ribeiro, J. Barata, M. Onori and A. Amado, "OWL Ontology to support Evolvable Assembly Systems", in 9th IFAC Workshop on Intelligent manufacturing Systems, 2008.
4) F. L. Bellifemine, G. Caire, and D. Greenwood, "Developing Multi-Agent Systems with JADE", Wiley, 2007.
5) J. Ferreira, L. Ribeiro, P. Neves, H. Akillioglu, M. Onori and J. Barata, "Visualization Tool to Support Multi-Agent Mechatronic Based Systems", in 38th Annual Conference on IEEE Industrial Electronics Society (IECON), Oct, 2012.

6) P. Ferreira, N. Lohse, M. Razgon, P. Larizza and G. Triggiani, "Skill-based Configuration Methodology for Evolvable Mechatronic Systems", in 38th Annual Conference on IEEE Industrial Electronics Society (IECON), Oct 2012.
7) R. S. Janka, "Specification and Design Methodology for Real- Time Embedded Systems", Kluwer Academic Publishers, 2002.
8) J. Barbosa and P. Leitao, " Simulation of Multi-Agent Manufacturing Systems using Agent Based Modeling Platforms", in 9th IEEE International Conference on Industrial Informatics (INDIN), 2011.
9) P. Vrba, "MAST: Manufacturing Agent Simulation Tool", in Emerging Technologies and Factory Automation, Proceedings ETFA IEEE Conference, 2003.
10) P. Mendhem and T. Clarke, "MAcSim: A Simulink Enabled Environment for Multi-Agent System Simulation", in Conference Proceedings IFAC, 2005.
11) S. Mahapatra, "A Hierarchical Approach to Modeling Agent-Based Systems in Simulink", in AIAA Modeling and Simulation Technologies Conference, 2012.
12) T. N. Qureshi, "Towards Model-Based Development of Self-Managing Automotive Systems", Licentiate Thesis, KTH, Royal Institute of Technology, 2009.
1) "Resource Allocation from Multiple Pools, Matlab R2014a Documentation", http://www.mathworks.se/help/simevents/examples/resourceallocation-from-multiple-pools.html, Last accessed April 2015.
13) T. N. Qureshi, D. J. Chen, L. Feng, M. Persson and M. Törngren, "On Mapping UML Models to Simulink/SimEvents: A Case Study of Dynamically Self-Configuring Middleware", Technical Report, Department of Machine Design, KTH, Royal Institute of Technology, 2009.
14) C. J, Sjöstedt, "Modeling and Simulation of Physical Systems in a Mechatronic Context", PhD Thesis, Department of Machine Design, KTH, Royal Institute of Technology, 2009.