Collaborative Systems of Systems Need Collaborative Design

John Fitzgerald, Jeremy Bryans, Peter Larsen, Hansen Salim

Collaborative Systems of Systems Need Collaborative Design

John Fitzgerald[1], Jeremy Bryans[1], Peter Gorm Larsen[2], Hansen Salim[1]

[1] School of Computing Science, Newcastle University, Newcastle upon Tyne NE1 7RU, United Kingdom
{John.Fitzgerald, Jeremy.Bryans, H.Salim}@ncl.ac.uk
[2] Department of Engineering, Aarhus University, Finlandsgade 22, DK-8200 Aarhus N, Denmark
pgl@eng.au.dk

Abstract. Advances in smart devices and networks are enabling convergence between systems of systems and cyber-physical systems in which computing elements interact closely with the physical environment. The development and maintenance of such systems should be inherently collaborative, crossing boundaries of constituent system ownership as well as semantic differences between engineering notations and disciplines. We present instances of model-based methods and tools that aim to bridge these gaps. Using a case study from smart grid control, we discuss the challenges in realising collaborative systems of systems that merit the reliance placed on them.

Keywords: Collaborative Systems of Systems, Cyber-Physical Systems, Formal Methods, Model-based Systems Engineering

1 Introduction

In areas as diverse as intelligent manufacturing or traffic control, advances in devices and networks are providing the basis of Cyber-Physical Systems (CPSs) that have potential to improve the quality of life of individuals and civic society. Such systems are assemblies of smart objects and pre-existing systems, many of them influenced by, and capable of influencing their physical environment. CPSs have potential to make spaces such as homes, factories and offices, smarter, more energy-efficient and comfortable because of the capacity to control their environment by making interventions in response to real-time data from internal and external sources.

In CPSs, the tight coupling of computing with the physical world found in embedded systems meets the independence, autonomy, heterogeneity and scale of Systems of Systems (SoSs). Although CPS technology is making significant progress, barriers to innovation remain, particularly the lack of modelling languages, tools, and associated foundations that support engineering of CPSs that are open and dynamic and yet dependable [1,2].

In this paper, we focus on the need to support the collaboration that is inherent in the engineering of SoSs and CPSs. This includes collaboration between the owners

and operators of otherwise independent constituent systems, and between the engineering disciplines involved in the design of a system in which computing elements interact with the physical world. We reflect on our experience in developing modelling frameworks for both SoSs and embedded systems, and consider whether together they hold out promise for supporting the design of trustworthy CPSs. In Section 2, we review the characteristics of SoSs that make their model-based design and maintenance a challenge, and discuss the realisation of a formal contract-based approach to model-based design of SoSs in the COMPASS project. In Section 3, we look to CPSs, discussing the implementation of cross-disciplinary co-modelling in Crescendo[1]. In Section 4, we present an illustrative example of co-modelling of a smart grid CPS. This leads to a brief discussion in Section 5 of the potential for bringing the technologies for SoS and co-modelling together.

2 Collaborative Modelling for SoSs

SoSs are composed of independent Constituent Systems (CSs) that are brought together in order to deliver a new service that the individual constituents could not offer separately. Classical examples include accident response as a result of collaboration between emergency services, or the delivery of an urban transport system composed of multiple providers.

Collaborative systems bring geographically dispersed teams together, supporting communication, coordination and cooperation [3], and are a significant enabling technology in both the creation and the operation of SoSs. In fact, a growing match between SoS and the discipline of collaborative networks has been noted [4]. The characteristics of SoSs induce requirements on the collaborative engineering processes and systems used in their design. We briefly consider some of the more important characteristics below.

2.1 Challenges of Model-based SoS Engineering

A SoS provides new *emergent behaviour* resulting from collaboration between constituent systems. However, unexpected emergence frequently has negative consequences [5], and a comprehensive approach to engineering SoSs must include provision to verify both the existence of positive emergent behaviour and the limits of unexpected and undesirable emergence.

Constituent systems are frequently *autonomous* in the sense that their behaviour is governed by their own goals rather than those imposed by the environment. Nevertheless, the reliance placed on the SoS's emergent behaviour necessitates an approach allowing constituents to offer behavioural guarantees to one another, sufficient to guarantee global SoS-level properties.

Constituent systems are *independently operated*, and would continue to function successfully if detached from the SoS. They are therefore likely to have much

[1] http://cresendotool.org

information about their day-to-day operation that is commercially sensitive. A collaborative engineering process should not require the exposure of this information. The guarantees required of a constituent system should not overly constrain its behaviour, but allow it freedom in deciding how to meet those guarantees. The ability to handle abstraction is therefore an important part of collaborative design.

In summary, model-based SoS engineering must support collaboration while allowing for constituent system autonomy and independence, and yet supplying sufficient information to permit verification of emergence [6].

2.2 Collaborative SoS Engineering: the COMPASS approach

In previous work, we proposed formal model-based methods for describing SoSs in terms of the interfaces offered by the constituent systems, with each interface defined contractually in terms of the assumptions it makes and the guarantees that it offers when those assumptions are satisfied [7]. In the COMPASS project[2], this approach has been realised in a modelling language (the COMPASS Modelling Language, CML [8]) that allows description of data, functionality and communication over an architectural model of the SoS given in SysML[3]. CML's semantic basis is extensible [9], making it possible to verify emergence by a range of techniques including simulation, model checking and proof, while the contractual approach permits the description of constituents without over-constraining each system's internals [10].

While a model-based approach based on contractual specification can be realised and supported by tools, the practical challenges of SoS engineering necessitate a further strengthening of tool support. To a large degree, SoS engineering depends on collaboration between integrators and owners of constituent systems. How can collaborators exchange information, agree on interfaces, prevent misunderstandings and ensure consensus, particularly when confidentiality may limit the extent to which collaborators permit access to interface specifications of their respective systems?

The need to support collaboration while respecting confidentiality is inherent in SoS engineering. Supporting it requires a step beyond beyond the kinds of Integrated Development Environment (IDE) developed for formal modelling and programming languages towards *Collaborative* Development Environments (CDEs) enabling negotiation and information hiding [11].

In COMPASS, tool support for model-based SoS engineering is based on the *Symphony* platform which integrates architectural and systems modelling tools based on SysML with CML (Fig. 1). In order to support collaborative development and (re-) negotiation of constituent system contractual interfaces, the framework has been extended with the concept of a *collaboration group*. Such a group is a collection of stakeholders (typically constituent system owners and integrators) who agree protocols for the development and evolution of interfaces developed for the SoS in question (Fig. 2), [12]. Within the collaboration group, rules govern the exposure of

[2] http://www.compass-research.eu
[3] http://www.omgsysml.org

model data and the iterative convergence on a mutually satisfactory set of reliances and guarantees. The analytic tools available in the COMPASS tool set are thus able to allow exploration and verification of alternative shared strategies, SoS architectures and allocations of responsibility to constituents.

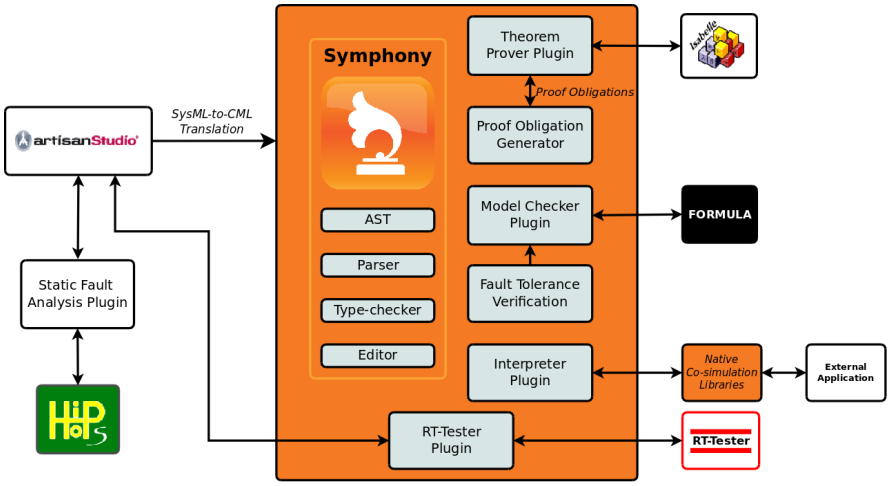

Figure 1: COMPASS tool support for formal contract-based SoS modelling

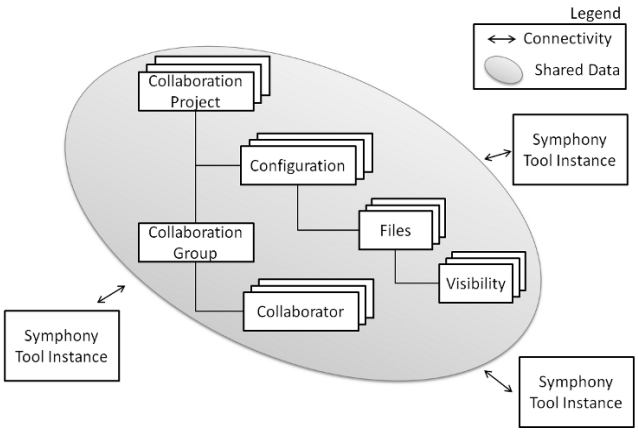

Figure 2: Enabling collaboration in Symphony

This contract-based approach has enabled the development of architectural models and modelling profiles for SoSs, and enabled the analysis of emergent behaviour in applications such as content-streaming networks [6].

3. Co-modelling of Cyber-Physical Systems

Our work on collaborative development of SoS models has focussed on the computing systems. However, CPSs demand semantically heterogeneous models. For example, there is a need to be able to allow developers to work together onelectro-mechanical and control elements most conveniently described in terms of Continuous Time (CT), as well as Discrete Event (DE) models of computing phenomena.

In our recent work, rather than developing a common hybrid modelling notation, we have sought to develop a common semantic basis allowing DE and CT models to be presented in established notations that would be familiar to engineers from different disciplines, but interfacing these (and their simulation environments), so that models can be developed and evaluated together in a common harness, rather than entirely separately (in a concurrent development process) or sequentially [13]. The resulting collaborative modelling (co-modelling) approach allows for DE and CT models in different notations (in our case VDM and 20-sim respectively) to be subjected to simulation in their own well-established simulation environments, but coordinated by a co-simulation engine (Fig. 3). The data and design parameters exchanged between the harnessed simulations are defined as a *contract* and the sequence of events and external inputs that are to take place during a co-simulation are termed a *scenario*.

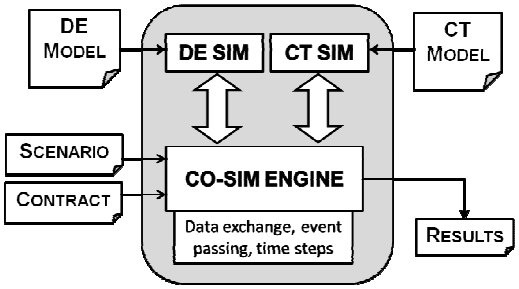

Figure 3: Collaborative simulation across disciplinary boundaries

The co-modelling approach allows the systematic exploration of both cyber and physical aspects of a system's design space. An example of this is considered below.

4 Example: Co-modelling of a Smart Grid

A *smart grid* uses digital technology to enable responsive and adaptive management of electricity generation, storage and consumption [14]. Data gathered from sensors on the grid enables forecasting, monitoring and response, including autonomous compensation for sensed failures. A smart grid is a CPS because of its computational integration with physical processes and has SoS characteristics through the integration of multiple constituent systems such as power generation and distribution facilities that may be independently owned and managed. A fault in such a system can have a

marked impact. For example, a software defect was identified as a significant factor in the US Northeast Blackout of 2003 [15].

To illustrate the co-modelling approach, consider a simple network containing two generators servicing domestic, industrial and hospital customers via a transmission system. Fig. 4 shows a CT model of the network developed in the 20-sim tool using notations and methods familiar to electrical and control engineers. The physical system is represented by multiple elements represented by icons, connected with lines that indicate power flow. Each icon stands for a system of differential equations describing the CT behaviour of the corresponding element. Generators are modelled as modulated voltage sources producing output that is stepped up by transformers for transmission, before being stepped down for consumers (represented by resistors). The "S" units are current sensors and switches. Under normal circumstances, Houses 1-6 are powered from Generator 2, and the other consumers from Generator 1. The Control Center (CC) represents the link, via the co-modelling contract, to the DE model of the control logic that reads the sensors and operates the switches. Space does not permit the description of the DE controller model, but this is represented in terms of data and functionality at an abstract level using VDM – a notation that has markedly greater abstraction than conventional programming languages, but the elements of which are familiar to software engineers.

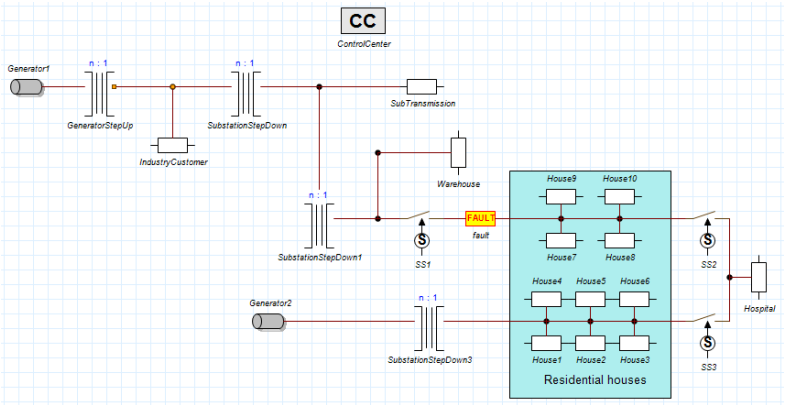

Figure 4: Smart grid CT model

We model a fault by defining a resistance unit that can be set to a low or high value under control from the DE side via its scenario (patterns for modelling CT and DE-side faults have been presented in [13]). Error detection and recovery is largely enabled by the sensed on current resulting from the sensors deployed along the smart grid. Careful design is required in the cyber-side part of the system, which must store and analyse collected sensor data.

In order to model and analyse the effects on the system as a whole of faults and attacks, it is important to be able to model defective or malicious behavior in both DE and CT formalisms. For example, a physical sensor defect naturally modelled in a CT formalism can result in corrupt data analysis that is best described in the DE formalism. Going in the opposite direction, cyber faults and attacks, such as defective

data transmission or SQL injection, are naturally modelled using a DE formalism, but their physical CT-side consequences are of real concern.

Co-modelling allows the specification of cyber-side faults and attacks in a notation suited for the purpose. For example, we can model a cyber-side fault in which sensor data is not read for a fixed period by a conditional in the DE controller model:

```
if time < 5E9 or time > 22E9 then updateSensordata();
```

In fact, when we combine this with the model of a physical fault introduced above, we observe the loss of the capacity to recover from the physical fault modelled CT-side. Since the database is not updated with the real-time sensor reading, the control centre (DE algorithm) uses the last data entry, and believes that things are running normally. Thus we are able to model the combination of DE-side and CT-side behaviours that might individually be tolerable but lead to emergent system-level failure. For example, in other studies, we have demonstrated power spikes resulting from a cyber-side attack that changes sensor values.

This ability to model and co-simulate collaboratively across discipline boundaries allows diverse teams of engineers to explore models rapidly in the early phases of their construction, in order to select optimal fault-tolerant designs.

5 Conclusions

We have examined the modelling of SoSs and simple CPSs. In the former, it is apparent that support for collaboration entails support for abstraction at constituent system interfaces, and a contractual approach has potential here. In the latter case, we have demonstrated the value of modelling techniques that allow the collaborative exploration of cyber-side and physical-side faults and attacks, and the consequences that they have *in combination* on the delivery of the overall system's emergent behaviour.

We would argue that the combination of co-modelling and a contractual approach to the modelling of interfaces of constituent systems has the potential for significant impact in engineering collaborative CPSs, although many challenges to fully realising this vision remain. A successful approach that permits the verification of emergent behaviour requires semantic support for multiple modelling paradigms. Our experience suggests that this should not supplant established formalisms, but should allow tool-level integration. Extensibility of semantic frameworks is important here, and COMPASS' UTP-based semantic framework is a first step.

We have examined only DE and CT formalisms, but one can imagine needing to integrate formalisms that support the description of features that are important to many CPSs, including the description of human behaviour, stochastic behaviour, and abstractions such as mobility and spatial characteristics. The integration of such modelling frameworks and appropriate tool support form a significant challenge.

Finally, our work has been motivated by dependability in the technical sense, but designing systems that command the less tangible property of trust from citizens

demands more radical approaches, and the possibility of human-centred design of SoSs and CPSs remains an open and intriguing area for potential research.

Acknowledgements. The work reported here has been supported by the EU FP7 projects COMPASS (Grant 287829) and DESTECS (Grant 248134), and by the EPSRC Platform Grant on Trustworthy Ambient Systems (EP/J008133/1). We are grateful to our many colleagues in all three project teams.

References

1. Broy, M. Engineering cyber-physical systems: Challenges and Foundations. In M. Aiguier, M., et al. (eds.). Complex Systems Design & Management, CSDM 2012, pp. 1–13. Springer (2013)
2. Lee, E.A. CPS Foundations. In Proc. 47th Design Automation Conference (DAC), pp. 737–742. ACM (2010)
3. Bafoutsou, G., Mentzas, G.: Review and Functional Classification of Collaborative Systems. In International Journal of Information Management 22. pp. 281-305 (2002)
4. Camarinha-Matos, L., Afsarmanesh, H.: Taxonomy of Collaborative Networks Forms, Final Report of FInES Task Force of Collaborative Networks and SOCOLNET – Society of Collaborative Networks, (2012)
5. Calder, M., Kolberg, M., Magill E. H., Reiff-Marganiec, S. Feature Interaction: A Critical Review and Considered Forecast. In Computer Networks, 41(1) pp 115-141. (2003)
6. Bryans, J., Fitzgerald, J., Payne, R., Kristensen, K., Maintaining Emergence in Systems of Systems Integration: a Contractual Approach using SysML. To appear in Proceedings of INCOSE 2014
7. Fitzgerald, J.S., Bryans, J.W., Payne, R.J.: A Formal Model-Based Approach to Engineering Systems-of-Systems. In Camarinha-Matos, L., Xu, L., Afsarmanesh, H. (eds.), Collaborative Networks in the Internet of Services - 13th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2012, IFIP Advances in Information and Communication Technology 380, pp53-62, Springer (2012)
8. Woodcock, J., Cavalcanti, A., Fitzgerald, J.S., Larsen, P.G., Miyazawa, A., Perry, S.: Features of CML: a Formal Modelling Language for Systems of Systems. In Proc.7th Intl. Conf. on System of Systems Engineering, IEEE (2012)
9. Woodcock, J.: Engineering UToPiA – Formal Semantics for CML. In Jones, C.B., Pihlajasaari, P., Sun, J. (eds.). FM 2014: Formal Methods, LNCS 8442, pp 22-41, Springer (2014)
10. Fitzgerald, J.S., Larsen, P.G., Woodcock, J.: Foundations for Model-based Engineering of Systems of Systems, in Aiguier, M. et al. (eds.) Complex Systems Design and Management, CSDM 2013, pp. 1-19, Springer, (2014)
11. Whitehead, J.: Collaboration in Software Engineering: A Roadmap. In 2007 Future of Software Engineering (FOSE'07), pp. 214-225, IEEE (2007)
12. Nielsen, C.B., Larsen, P.G.: Collaborative Formal Modeling of System of Systems. In Proc. 8th Annual IEEE Systems Conference (SysCon), pp. 154-161, IEEE (2014)
13. Fitzgerald J.S., Larsen P.G., Verhoef M.H.G., (eds.): Collaborative Design for Embedded Systems: Co-modelling and Co-simulation. Springer (2014)
14. Gellings, C.W.: The Smart Grid: Enabling Energy Efficiency and Demand Response. Lilburn: The Fairmont Press, Inc. (2009)
15. U.S.-Canada Power System Outage Task Force. Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations (2004)