Efficient Incident Handling in Industrial Automation through Collaborative Engineering

Jan Olaf Blech*, Ian Peake*, Heinz Schmidt*
Mallikarjun Kande**, Akilur Rahman**, Srini Ramaswamy§, Sudarsan SD**, Venkateswaran Narayanan**
* RMIT University, Melbourne, Australia
** ABB Corporate Research, Bangalore, India
§ ABB Power Systems and Power Generation R & D, USA

***Abstract*—We present our monitoring and decision framework for collaborative engineering for globally distributed operation, support, maintenance, and services for industrial automation. The framework provides relevant information to plant operators, engineers, staff and stakeholders to support the handling of incidents, based on semantically-appropriate factors such as personnel skills, physical location of affected equipment and dependencies between plant elements. We discuss the proposed application and present the architecture and implementation. Based on incoming events the framework selects, aggregates and displays information automatically for human processing possibly at distant control centres. For example an alarm in a manufacturing facility can trigger the display of relevant information on multiple devices such as workstations, tablets, or large control-room screens to supervisors and experts. Devices can be potentially in different locations and can comprise different visualization capabilities. The core of our framework uses semantic models and formal methods-based techniques to aggregate and process this information.**

I. Introduction

Over more than two decades, multinational projects are driven by global collaborative engineering, forming dynamic socio-technical networks of enterprises and projects with access to the best expertise, organisational/enterprise support, facilities, compute and data services, often distributed across several points on the globe. Our collaborative engineering work integrates with *virtual enterprise architecture*. Here, engineering domains deal with enterprise architecture, driving architectural transformation in the presence of global distribution, contractual and ad-hoc coalitions of multiple enterprises to achieve specific productivity and project goals, leveraging shared resources, skills and competencies of the members of the coalition [25] while preserving quality of software and systems architectures [32]. Especially for large-scale industrial projects, additional drivers are:

- the need to continually integrate and disintegrate technologies as multinationals merge or spin out subcompanies through company acquisitions and sale;
- compliance with changing legislation and regulation in different jurisdictions of the globally networked company or groups of companies;
- increasing outsourcing at the enterprise level, balancing cost reductions (in particular in wages) with additional time and organisational costs incurred;
- software-intensity enabling flexible customization and continuous service improvement (post deployment);
- transitioning from product-centred to service-centered business models (approximately 70% of GDP in OECD countries is now service-based).

Dealing with collaboration and virtual enterprise architecture over significant distances, especially in global software engineering, the provision of engineering services and software, its testing and monitoring and generally service support, is essential for end-to-end cost savings, large-scale technology deployment, repurposing and subsequent business services. Megatrends such as the *internet of things* (IoT), *mobility* (devices in the hand of engineers or mounted on mobile infrastructure and equipment), and *cloud* provide massive opportunities for disruptive changes in services and associated gains in market share. However, there are also significant obstacles and demands for exploring pragmatic 'low-hanging fruit' approaches, with the aim to scope potential costs and benefits, weaknesses and strength of promising combinations of existing technology and size up opportunities and threats.

Our collaborative engineering project aims at facilitating the exchange of data and knowledge for remote plant operation, services and maintenance. Here, we focus on a decision support framework, that provides relevant information to plant operators, engineers, other staff and stakeholders that support the handling of events. In industrial automation events comprise *alarms* issued by a control system such as a SCADA (supervisory control and data acquisition) system which are based on sensor values deviating from a pre-defined range or manual triggering. The decision support relies on models encapsulating the semantics of system components and ontologies. As shown in Figure 1, collaboration can be distributed among different sites including plants, and operation centers. Sites from different domains (e.g., manufacturing and mining), from different operators and owners and from distant areas can be connected through the collaborative engineering framework.

Our decision support framework presented here integrates with the Microsoft SharePoint platform and the .Net framework. Decision support is based on incoming events and realized using a monitoring and troubleshooting unit which decides on relevant information from additional sources and

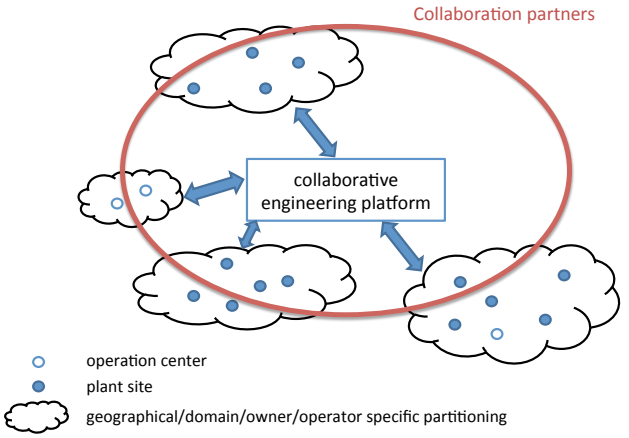

Fig. 1. Distributed collaboration

solving strategies. Semantic models and ontologies are encoded using logic based formal descriptions. Decision support uses the semantic models and is realized using constraint solving based on formal methods. The results of constraint solving are used to generate display information encoded in XML which are then used to present relevant information to plant operators, engineers and other stakeholders. Our spatial constraint solver BeSpaceD is used for supporting semantic model based decisions [5], [6]. The visualization information is displayed using different devices. In particular, we are using our VxLab [7] facility which offers multi-screen visualization.

First ideas on collaborative engineering support have been published by us in an initial vision paper [10]. Here, we present a refined version of our idea and describe 1) a more detailed architecture, 2) the implementation of a framework prototype, 3) the creation of a language for controlling the display of event related content, 4) the positioning of the framework into a broader context as novel contributions in this paper.

Overview

Related work is discussed in Section II Section III presents a motivating example use case. We discuss our BeSpaceD tool and spatial reasoning in Section IV. Our collaborative engineering framework is presented in Section V. Section VI describes visualization for different devices. Section VII presents a conclusion.

II. Related Work

Existing commercial collaboration frameworks such as Microsoft SharePoint, Dassault Système's Enovia [20] and Delmia [19] concentrate on providing solutions for the exchange of documents such as texts. These frameworks are currently used to support collaboration between and within organisations in business and in the operations control context, including industrial operations sites.

Early academic approaches and tools for collaborative engineering are described in [17] and [27]. Sharing and collaborative interaction on documents and other sources is presented. Most of the ideas provided by these tools have found entry into commercial collaboration frameworks. Academic approaches comprise examinations such as the impact of collaborative engineering on software development (see, e.g., [11]). Moreover, another study collects software challenges for the development of ultra-large scaled systems [30], in which collaboration is limited to the development phase.

Assigning semantics to distributed documents and information has gained popularity in the context of the semantic web [4]. This can provide a basis for collaboration between different industrial sites. Ontology-based approaches to collaborative engineering based on semantic web technology are described in [35]. The ComVantage project [31] is developing a mobile enterprise reference framework for future internet operability. Semantic annotations of data are taken into account to some extent. Applications in industrial automation exist.

In the social network context [16] of SharePoint the analysis of collaboration relations between different participants is of relevance. Analysing these relations can serve as a basis for efficiency improvement, by identifying candidates for working groups. Furthermore, it can be used in approaches to resolve resource conflicts and protecting confidentiality.

Models and GIS: Semantic models are an important basis for automatically supporting collaboration. Data models for cyberphysical infrastructure in construction, plant automation and transport – domains serving as a basis for adequate production plant or mining operation modeling – have been studied in the past. Many of the existing real-world applications are aligned towards a geometric representation of components and are typically based on so-called 2.5 dimensional GIS (Geographic Information System) representations where the 3rd dimension $z = f(x, y)$ is represented as a function f of the 2 dimension x and y coordinates. This [2] can limit the ability to use these models as a basis for geometric, topological and information retrieval. True three dimensional modeling is far from common practice [33]. In our models, we do not limit ourself to a particular geometric representation, coordinate or dimension system. Possible extensions of interest include consideration of standards such as the Web 3D Services and the Sensor Web Enablement Architecture of the Open Geospatial Consortium [1], visualisation and decision support [36] and efficient data structures for fast meta reasoning and presenting subproblems of choice to specialist solvers as provided by and used within BeSpaceD.

Logic for Space: Logic for space is used as a basis for our semantic models of production plants. The handbook of spatial logic [1] discusses spatial logics, related algebras and applications. The book covers a large spectrum not only limited to computer science. A formalism based on ideas of process algebras has been introduced in [13] and [14]. It is used for describing and reasoning about spatial behavior. Process algebras provide a precise formal semantics definition and are aimed towards the specification of highly parallel systems. Here, disjoint logical spaces are represented in terms of expressions by bracketing structures and carry or

[1] http://www.opengeospatial.org

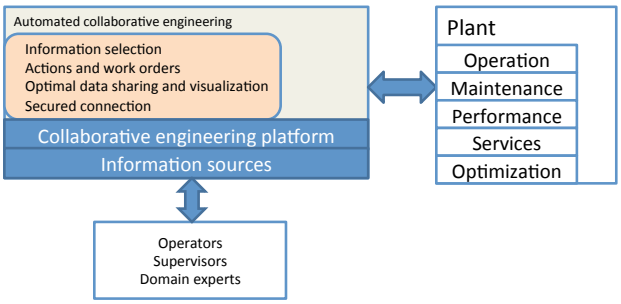

Fig. 2. Collaborative engineering from a user-perspective

exchange concurrent process representations. Model checking for process algebras such as spatial behavioral specifications is presented in [15]. A graph-based technique for the verification of spatial properties of finite π-calculus fragments is introduced in [23]. Results on spatial interpretations are presented in [28]. Many aspects of spatial logic are in general undecidable. As a notable exception, a quantifier-free rational fragment of ambient logic (corresponding to regular language constraints) has been shown to be decidable in [37]. Special modal logics for spatio-temporal reasoning go back to the seventies. The Region Connection Calculus (RCC) [3] includes spatial predicates to indicate spatial separation of regions. Among others RCC features predicates indicating that regions do not share any points at all, points on the boundary of regions are shared, internal contact where one region is included and touches on the boundary of another from the inside, proper overlap of regions, and proper inclusion. Furthermore, [3] features an overview of the relation of these logics to various Kripke-style modal logics, reductions of RCC-style fragments to a minimal number of topological predicates, their relationship to interval-temporal logics and decidability.

Previously we have published our initial ideas for collaborative engineering in [10]. Our spatial constraint solving framework is described in [5], [6]. Work on our VxLav visualization facility can be found in [7]. Our framework for collaborative requirements engineering: C-FaRM is described in [26]. Other background work on technologies for collaboration can be found in [12], [24].

III. Example Use-Case

Figure 2 shows the collaborative engineering framework from a user perspective. Humans, such as plant operators or experts interact directly with the framework. The framework gathers data from and interacts with plant operations.

Here, to illustrate our framework we are using an example use-case. Figure 3 shows our framework responding to an event, such as an alarm triggered by a machine malfunction.

We consider the following concrete event sequence (use-case) and show the processing steps in the framework (cf. [10]):

1) A sensor provides data indicating a machine malfunction in a remote plant. Based on the information available we check the confidence by investigating historical data from the sensor and by using data collected from nearby sensors. This is used to decide whether there is indeed a failure.
2) If there is a failure, we generate an alarm which is provided to our monitoring and troubleshooting unit. Together with the actual alarm indicating the type (here: a machine malfunction), we can provide additional data.
3) Based on this input, our goal is to provide the information to staff, stakeholders, experts, and/or engineers. Information has to be provided in a concise way. This means, that we can face situations, where many alarms arrive in a short time and information for display has to be filtered accordingly so that humans are not overburdened with information. This is especially true, if the source of alarms is of an external nature and effecting multiple devices at the same time such as a lightning strike in a power substation. The involved humans can be chosen by capabilities, availability or proximity, in case a physical investigation is necessary or a remote support is sufficient. Using our BeSpaceD based reasoning, our collaboration platform can automatically match experts to the situation and offer resource conflict resolution while alerting them or a manager. Monitoring and troubleshooting can take additional information into account, e.g., information provided in databases including semantic models of plants and processes and real-time information from streaming sources. In this use-case, semantic models provide information on states of machinery, possible dangers, possible interactions, physical locations, possible effects on the surrounding area.
4) Using the results of the monitoring and troubleshooting unit, we generate incident relevant information for display to humans. We use commands that trigger changes to display information on mobile, devices, normal workstations or large scale visualization facilities.

Typical information displayed comprises profiles and other data stored in SharePoint as well as maps. The SharePoint-based data is triggered in browser windows opened by our monitoring and troubleshooting unit. It can than be handled interactively, e.g., our monitoring and troubleshooting unit triggers the display of an editable collaborative document on multiple devices in different sites.

IV. BeSpaceD and Spatial Reasoning

BeSpaceD [5], [6] is used in our collaborative reasoning framework for spatial reasoning and verification. BeSpaceD is a constraint solving and non-classical model checking framework. Here, we use it to dynamically and statically decide on consequences and relevant information of an event / alarm occurring in a system and semantically interpret (series) of alarms occurring in a system.

We have applied BeSpaceD in a number of different projects (e.g., [21], [22]). A special emphasise of BeSpaceD is on dealing with models that comprise a large amount of time and space based specifications.

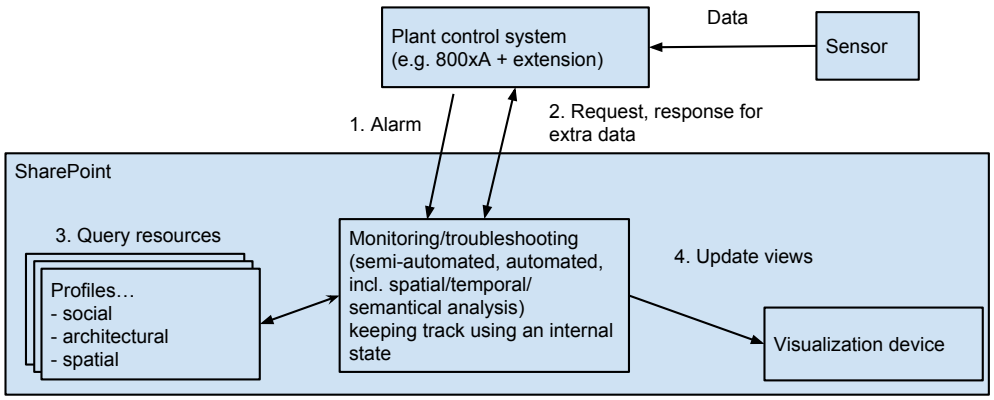

Fig. 3. Collaborative engineering use-case and the architecture [10]

The BeSpaceD framework comprises

1) a modeling language focusing especially on space and time, and
2) a library to reason on models. Library functions comprise state-space exploration, abstraction and reduction to checks properties of the models expressed in the BeSpaceD modeling language.

BeSpaceD based constraint solving can be done by either calling library functions or writing customized checking procedures using the existing library functions.

Our BeSpaceD modeling language allows the time or automata based behavioral descriptions for entities in combination with their spatial (coordinates or topological) characteristics. These can be used to describe availability areas and schedules, capabilities and additional aspects such as events and states.

a) BeSpaceD based constraint solving: BeSpaceD-based checking may be done by using the series of steps sketched below to perform the checking as shown in Figure 4. BeSpaceD functions and language elements are combined.

- In a first preprocessing step, an algorithm may be used to identify large scale constraint solving goals. For example static components that are separated in space and do not interact with each other may be checked independently. The behavior of mobile components may also be checked independently if they are only acting in a local area in space or time. Properties may only regard a local area and therefore only the behavior of components that act within this local area has to be taken into account. Best practices for this approach can be scenario and domain specific. Furthermore, open parameters of properties and models are instantiated.
- Depending on the constraint solving goal in a next step, we can create abstractions of behavior: *invariants* for all relevant components. This can be done by unfolding predicates in the logical description of the original semantic model described in the BeSpaceD modeling language in a safe way. For example a description indicating a spatial incident in around a circle on a plane for a time interval may be unfolded into a conjunction of specifications indicating the spatial incident for several points in time and space that cover the circle and its time interval.
- We use these invariants to generate verification conditions in a next step. Verification conditions can be checked by separate highly specialised tools like SAT [2] and SMT solvers (e.g., Yices [18] or z3 [29]).
- The last step collects the results of used verification tools and presents an overall results. Optionally, we refine invariants or verification conditions, if the result does not satisfy our needs.

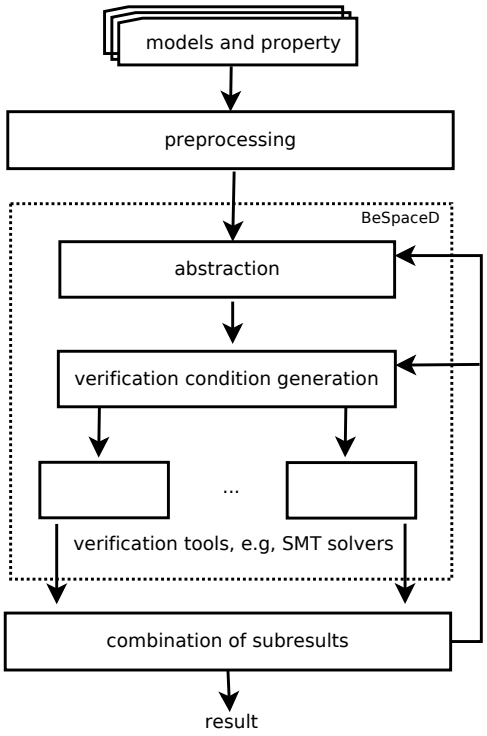

Fig. 4. BeSpaceD use for deciding a constraint problem [5]

[2]e.g., by using Sat4j: http://www.sat4j.org/

Computation of invariants can be done in parallel in many cases. Verification conditions never depend on each other, so they can be always checked in parallel.

Results of the checking comprise binary answers (yes/no) indicating whether a property holds for the model(s), finding of solutions, e.g., a property that needs to hold on a model, iterative finding of different properties as solutions, and prioritization of different solutions based on criteria specified in the property and model.

b) Models for BeSpaceD: Two possibilities exist for creating semantic models in BeSpaceD:

1) Models may be manually written directly using the BeSpaceD modeling language.
2) Generation of behavioral models from code is shown in Figure 5. Here, BeSpaceD descriptions are created by executing customized code pieces. In the shown excerpt, a trajectory is created indicating the position of moving elements in a factory depending on time.

V. Our Collaborative Engineering Framework

The architecture of our framework comprises ingredients to collect events and process and display information appropriately.

A. Framework Architecture Overview

Figure 6 shows the implementation of the core of our collaboration support: the monitoring and troubleshooting unit and its connection to visualization facilities. The monitoring and troubleshooting unit takes events and passes them to event specific handlers. The event specific handlers are responsible for triggering the display of information to human experts, collaborators, stakeholders. Event specific handlers share a global state. This state is used to share information between event specific code and for keeping track of the event history. Triggering the display of information is done by emitting XML code that is interpreted by a visualization manager. The visualization manager triggers the display of the selected information in device specific ways, e.g., for mobile devices, workstations or our large screen visualization facilities (cf. Figure 7).

Figure 8 shows the internals of the monitoring and troubleshooting unit. Events are collected and a timestamp can be added. In a second step events are sorted and added to queues for event specific handling. The event specific handling can be parallelised. Event specific handlers can exchange and store information thereby realizing the shared state from Figure 6. Finally, the event specific code is also responsible for emitting the XML code.

B. Handling Events

We invoke our spatial reasoning and decision checker BeSpaceD (cf. Section IV). To give a look and feel of a semantic carrying model, used to support the event handling, Figure 9 shows a small excerpt of a factory hall model. In the first line, the spatial outline of the factory hall is described. The next 5 lines define the position of fire sensors. The next lines

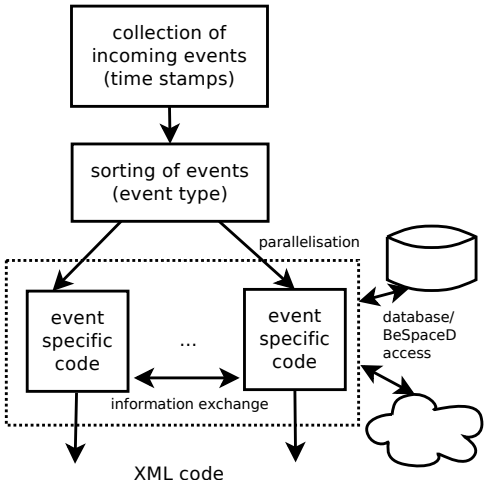

Fig. 8. Monitoring and troubleshooting

formalize the areas in the factory hall (circular, hence, the radius construct) in which a fire can be detected with a given probability. The last line describes the availability of a fire extinguisher and its position.

We can use the presented spatial model to statically check properties such as:

Is there a complete coverage of the factory hall with fire sensors?

Furthermore, we can use it dynamically, e.g., in combination with a reported alarm. Here, we may be interested in questions such as:

What is the distance between the location of the alarm indicating sensor and the fire extinguisher?

The models are combined with dynamic information to decide coverage and impact questions based on ongoing events and provide appropriate information.

C. Generating Visualization Output

Our aim is to support event- and stakeholder-specific delivery of notification elements of scalable complexity, ultimately up to custom applications with interactive content or more specific collaboration tools. Depending on the event, its location and the relevant stakeholders, different notifications and representations may be needed. Our XML format provides a notation for expressing and initiating notifications which include abstraction of stakeholder and/or location. Based on XML, notifications are delivered using visualization managers. Providing a look and feel, Figure 10 contains example XML output for visualization. Each command triggers the display of an individual window. The arguments allow the specification if the content. The first command displays an image with annotations `display` command featuring graphical annotations. The next two commands trigger the display of google map views with coordinates and zoom factor. The last command triggers a live camera view. Other commands feature the

```
for (i <- 500 to 600) {
  inv2b ::= (IMPLIES(TimeStamp (i),
      AND(
               OccupySegment(300,200,300 + (150.0*Math.sin(i*0.1)).toInt,
                   200 + (150.0 * Math.cos(i * 0.1)).toInt,10),
       (AND(
               OccupySegment(
                   300 + (150.0 * Math.sin(i*0.1)).toInt,
                   200 + (150.0 * Math.cos(i * 0.1)).toInt,
                   300 + (150.0 * Math.sin(i*0.1)).toInt+ (50.0 * Math.sin(i*0.5)).toInt,
                   200 + (150.0 * Math.cos(i * 0.1)).toInt + (50.0 * Math.cos(i*0.5)).toInt ,4),
              OccupySegment(
                  500,400,500 + (120.0 * - Math.sin(i*0.2)).toInt,
                  400 + (120.0 * - Math.cos(i * 0.2)).toInt,12))  ))));
}
```

Fig. 5. Code generating behavior

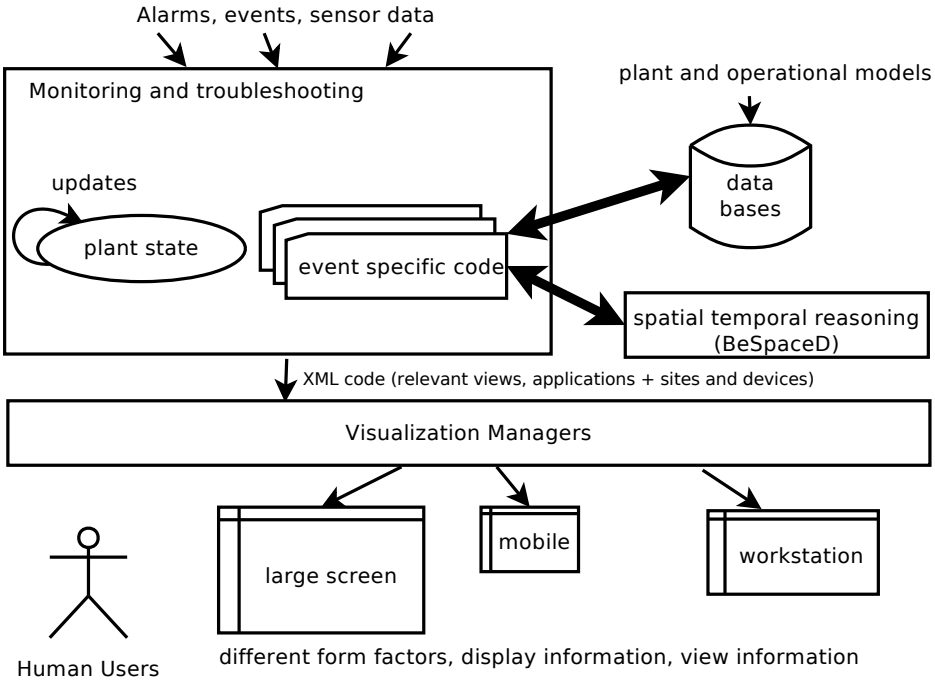

Fig. 6. The monitoring and troubleshooting unit and visualization

Fig. 7. Our multi-screen visualization [9]

```
 def FactoryHall = IMPLIES(Owner("FaxctoryHall"),OccupyBox(50,50,250,150))

def FireSensor1 = IMPLIES(Owner("FireSensor1"),OccupyPoint(80,100))
 def FireSensor2 = IMPLIES(Owner("FireSensor2"),OccupyPoint(120,100))
def FireSensor3 = IMPLIES(Owner("FireSensor3"),OccupyPoint(160,100))
def FireSensor4 = IMPLIES(Owner("FireSensor4"),OccupyPoint(200,100))
def FireSensor5 = IMPLIES(Owner("FireSensor5"),OccupyPoint(230,100))

def fsdetrange = BIGAND (
IMPLIES(AND(Event("FireDetection"),Prob(1.0)),OccupyCircle(80,100,10))::
IMPLIES(AND(Event("FireDetection"),Prob(.95)),OccupyCircle(80,100,90))::
IMPLIES(AND(Event("FireDetection"),Prob(1.0)),OccupyCircle(120,100,10))::
IMPLIES(AND(Event("FireDetection"),Prob(.95)),OccupyCircle(120,100,90))::
IMPLIES(AND(Event("FireDetection"),Prob(1.0)),OccupyCircle(160,100,10))::
IMPLIES(AND(Event("FireDetection"),Prob(.95)),OccupyCircle(160,100,90))::
IMPLIES(AND(Event("FireDetection"),Prob(1.0)),OccupyCircle(200,100,10))::
IMPLIES(AND(Event("FireDetection"),Prob(.95)),OccupyCircle(200,100,90))::
IMPLIES(AND(Event("FireDetection"),Prob(1.0)),OccupyCircle(230,100,10))::
IMPLIES(AND(Event("FireDetection"),Prob(.95)),OccupyCircle(230,100,90))::
Nil)

def FireExtinguisher = IMPLIES(Owner("FireExtinguisher"),OccupyPoint(100,100))
```

Fig. 9. BeSpaceD model

```
<output>
      <command device='govlab' type='display' profile='machineprofile1'></command>
      <command device='miniwall' type='display' profile='alice'></command>
      <command device='miniwall' type='display' profile='charles'></command>
      <command device='incrc' type='event' catagory='correctcoverage' id='1001'></command>
      <command device='auabb' type='view' image='substation9.jpg'
            rectx='300' recty='500' rectw='150' recth='100' text='hello world'
            txtx='100' txty='50'></command>
      <command device='riotinto-pilbara' type='map' lat='-37.800114'long='144.9671113'
            zoom='15z'></command>
      <command device='riotinto-mel' type='earth' lat='-37.800114'long='144.9671113'
            height='100m'>
      </command>
</output>
```

```
<output>
   <command type='composite' image='substation9.jpg'>
      <display type='rect' x='300' y='500' w='150' h='100'></display>
      <display type='text' text='substation incident' x='100' y='50' color='blue'></display>
   </command>
   <command type='map' lat='-37.800114' long='144.9671113' zoom='15z'></command>
   <command type='earth' lat='-37.800114' long='144.9671113' height='100m'></command>
   <command type='display' profile='camera_view'></command>
</output>
```

Fig. 10. XML output containing visualization information (device independent and device specific code)

display of staff and machine profiles as well as opening webbrowsers with a given URL.

The XML code is than provided to a visualization manager.

VI. Visualization using different Form Factors

Stakeholder location (physical and network) determines not only availability, but also their available forms of display and modes of interaction: from land line telephone through smartphones or tablets to control rooms and virtual and augmented reality environments. Visualization managers use standard methods such as web services to deliver notifications. A visualization manager may notify a client on the relevant device, nominating an address to be used as a callback to the visualization manager service.

For example, our collaborative engineering prototype uses a video wall visualization manager (WViM) to deliver notifications when the specified receiver is associated with a video wall in our VxLab [7], [8], [9], including our Global Operations Visualization (GOV) Lab at RMIT. The GOV lab includes an approximately 7m x 3m video wall enabling research in distributed and remote software engineering services such as remote testing for industrial automation. Our video walls use the SAGE middleware [34]. Each video wall in VxLab has an associated web service interface which supports actions such as starting a local application on the video wall, e.g., a web client or video conference. The WViM starts a web client on the relevant video wall/s, nominating a web service address to be used as a callback to the WViM. A generic or specific page may be delivered depending on the stakeholder,

stakeholder location, and event. Currently the prototype uses SharePoint to serve these web pages.

VII. Conclusion

This paper presented an overview on the existing implementation of our collaborative engineering framework. Collaborative engineering aims at facilitating the exchange of data and knowledge for remote plant planning, deployment, operation, services and maintenance, to improve quality of service and expertise while reducing costs through cooperation and outsourcing. Here, we focused on a decision support framework. The focus is on providing relevant information on incoming events to plant operators, engineers, and other staff and stakeholders, possibly across company, country and jurisdiction boundaries.

So far a first prototype has been implemented which is able to successfully demonstrate the concept. We did not conduct detailed load or performance tests on large scale realistic data and usage scenarios yet. The establishment of realistic plant models for larger industrial facilities providing us with test scenarios is an independent research task, is subject to future work and requires close interactions with plant operators. A second direction for future work is targeting even more decoupling of the architecture and the migration of parts of the used software tools into a distributed (cloud-based if compliance constraints can be met) environment.

References

[1] M. Aiello, I. E. Pratt-Hartmann, and J. FAK van Benthem, eds. Handbook of spatial logics. Springer, 2007.

[2] M. Apel. A 3D geological information system framework. Geophysical Research Abstracts, vol. 7, European Geosciences Union, 2005.

[3] B. Bennett, A. G. Cohn, F. Wolter, M. Zakharyaschev. Multi-Dimensional Modal Logic as a Framework for Spatio-Temporal Reasoning. Applied Intelligence, Volume 17, Issue 3, Kluwer Academic Publishers, November 2002.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific american 284, no. 5 (2001): 28-37.

[5] J. O. Blech and H. Schmidt. BeSpaceD: Towards a Tool Framework and Methodology for the Specification and Verification of Spatial Behavior of Distributed Software Component Systems. In *arXiv.org*, http://arxiv.org/abs/1404.3537, 2014.

[6] J. O. Blech and H. Schmidt. Towards Modeling and Checking the Spatial and Interaction Behavior of Widely Distributed Systems. In *Improving Systems and Software Engineering Conference*, 2013.

[7] J. O. Blech, M. Spichkova, I. Peake, and H. Schmidt. Cyber-Virtual Systems: Simulation, Validation & Visualization. In *Evaluation of Novel Approaches to Software Engineering*, 2014.

[8] AICAUSE Research Centre - Facilities `http://www.rmit.edu.au/browse;ID=ekm9dvmk1mqi`, 2014.

[9] J. O. Blech, M. Spichkova, I. Peake, H. Schmidt. Cyber-Virtual Systems: Simulation, Validation & Visualization. Evaluation of Novel Approaches to Software Engineering. Lisbon, 2014.

[10] J. O. Blech, I. Peake, H. Schmidt, M. Kande, S. Ramaswamy, Sudarsan SD., and V. Narayanan. Collaborative Engineering through Integration of Architectural, Social and Spatial Models. *Emerging Technologies and Factory Automation (ETFA)*, IEEE Computer, 2014.

[11] G. Booch and A. W. Brown. Collaborative development environments. Advances in Computers, 59 (2003): 1-27, Academic Press, 2003.

[12] A. Cabani, S. Ramaswamy, M. Itmi, J.P. Pécuchet, PHAC: an Environment for Distributed Collaborative Applications on P2P Networks, International Journal of Intelligent Control and Systems, Vol 12, #3, Sept 2007.

[13] L. Caires and L. Cardelli.A Spatial Logic for Concurrency (Part I). Information and Computation, Vol 186/2 November 2003.

[14] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part II). Theoretical Computer Science, 322(3) pp. 517-565, September 2004.

[15] L. Caires and H. Torres Vieira. SLMC: a tool for model checking concurrent systems against dynamical spatial logic specifications. Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2012.

[16] R. Cross, S. P. Borgatti, and A. Parker. Making Invisible Work Visible: USING SOCIAL NETWORK ANALYSIS TO SUPPORT STRATEGIC COLLABORATION. California management review 44, no. 2 (2002)

[17] M. R. Cutkosky, J. M. Tenenbaum, and J. Glicksman. Madefast: collaborative engineering over the Internet. Communications of the ACM 39.9 (1996): 78-87.

[18] B. Dutertre, L. De Moura. The yices smt solver. Tool paper at http://yices.csl.sri.com/tool-paper.pdf, 2006.

[19] DS DELMIA V6R2013x – Fact Sheet: 3DEXPERIENCES of Global Production Systems for all stakeholders in the extended supply chain. Dassault Systèmes 2013.

[20] ENOVIA V6R2013x – Fact Sheet. Dassault Systèmes 2013

[21] F. Han, J. O. Blech, P. Herrmann and H. Schmidt. Towards Verifying Safety Properties of Real-Time Probabilistic Systems. Formal Engineering approaches to Software Components and Architectures, 2014.

[22] F. Han, J. O. Blech, P. Herrmann, H. Schmidt. Model-based Engineering and Analysis of Space-aware Systems Communicating via IEEE 802.11. COMPSAC, IEEE, 2015. *to appear*

[23] F. Gadducci and A. Lluch Lafuente. Graphical Verification of a Spatial Logic for the π-calculus." Electronic Notes in Theoretical Computer Science 154.2, 2006.

[24] S. J. Geoghegan, G. McCorkle, C. Robinson, J. Brown, G. Fundyler, S. Ramaswamy, M. Tudoreanu, R. Seker and M. Itmi, A Multi-Agent System Architecture for Cooperative Maritime Networks 3rd Annual IEEE International Systems Conference, Vancouver, Canada, March 2009.

[25] A. Goel, H. Schmidt, D. Gilbert: Towards formalizing Virtual Enterprise Architecture. Proc. 13-th IEEE EDOC 2009, pp. 238-242, IEEE, 2009.

[26] S. Ghosh, A. Dubey, S. Ramaswamy. C-FaRM: A Collaborative and Context Aware Framework for Requirements Management, 4th International Workshop on Managing Requirements Knowledge, 19th IEEE International Requirements Engineering Conference, August 30th 2011, Trento Italy.

[27] McGuire, J. G., Kuokka, D. R., Weber, J. C., Tenenbaum, J. M., Gruber, T. R., & Olsen, G. R. (1993). SHADE: Technology for knowledge-based collaborative engineering. Concurrent Engineering, 1(3), 137-146.

[28] D. Hirschkoff, É. Lozes, D. Sangiorgi. Minimality Results for the Spatial Logics. Foundations of Software Technology and Theoretical Computer Science, vol 2914 of LNCS, Springer, 2003.

[29] L. De Moura, N. Bjørner. Z3: An efficient SMT solver. In Tools and Algorithms for the Construction and Analysis of Systems (pp. 337-340). Springer, 2008.

[30] L. Northrop, P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman et al. Ultra-Large-Scale Systems-The Software Challenge of the Future. (2006).

[31] A. Salmen et al. ComVantage: Mobile Enterprise Collaboration Reference Framework and Enablers for Future Internet Information Interoperability. Future Internet, vol. 7858 of LNCS, Springer 2013.

[32] H.W. Schmidt, I.D. Peake, J. Xie, I. Thomas, B. Krämer, A. Fay, P. Bort: Modelling predictable component-based distributed control architectures. Proc. WORDS 2003, IEEE, pp. 339-346, 2003.

[33] G. Smith and J. Friedman. A Technology Whose Time Has Come. Earth Observation Magazine, November 2004.

[34] L. Renambot, et al. "SAGE: the scalable adaptive graphics environment." Proceedings of WACE. Vol. 9. No. 23. 2004.

[35] Y. Sure, M. Erdmann, J, Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. Springer Berlin Heidelberg, 2002.

[36] C. Weaver, D. Peuquet, A. M. MacEachren. STNexus: An Integrated Database and Visualization Environment for Space-Time Information Exploitation. http://www.geovista.psu.edu/publications/2005/Weaver_ARDA_05.pdf, 2005.

[37] S. Dal Zilio, D. Lugiez, C. Meyssonnier. A logic you can count on. Symposium on Principles of programming languages, ACM, 2004.