2013 IEEE Third International Conference on Cloud and Green Computing

A Fresh Look At Modeling Distributed Reactive Systems

Roland Moser, Schahram Dustdar
Vienna University of Technology
Vienna, Austria
e0125617@student.tuwien.ac.at,
dustdar@infosys.tuwien.ac.at

Johannes Gutleber, Luciano Orsini
CERN
Geneva, Switzerland
johannes.gutleber@cern.ch,
luciano.orsini@cern.ch

***Abstract*—Scalable, distributed reactive systems require a scalable control infrastructure. Being able to model such an infrastructure is a prerequisite for putting it into place. In current systems, finite state machine based approaches are prevailing. They are, however, limited by their nature and hamper further improvements. Extensions like statecharts have emerged to alleviate those shortcomings. Despite all this work, the proposed improvements do not live up to the requirements of continuously operating distributed systems. Therefore we look for alternatives that break with the underlying state-based model. Workflow systems recently became mature enough to be considered for use cases going beyond small-scale business applications. The concept on which they build is promising to scale to large systems. This paper compares various modeling technologies using the real-world scenario of distributed data acquisition systems as we find them in currently operating high-energy physics installations at the Large Hadron Collider at CERN. The outcome of our qualitative evaluation will show that current finite state machine compared to workflow based approaches have a bigger gap between specification and executed processes. Consequently workflow based approaches are better suited for the use case at hand and require less custom software extensions to reduce the remaining gap.**

Keywords—distributed systems; reactive systems; state machines; state charts; ECA rules; workflows; web services

I. Introduction

With ever growing amounts of data to be processed and with systems that are built rather from composing services than designing them at the drawing board, mastering system complexity becomes a challenge. To avoid inconsistencies at runtime, the gap between specification and running processes must be kept small [*1*]. Ideally processes are a result of directly executing models. In this context we evaluate different specification models.

Models must be able to address diverse problems. Installations need to scale up to several thousands of interconnected nodes, include heterogeneous, interdependent services from a number of different vendors. Tasks contain real-time paths and most important do not have a single flow of execution. These systems are ever-running distributed applications that cooperate autonomously to process data [*2*].

Systems of that kind prevail in environmental monitoring [*3*], power distribution [*4*], plant control systems [*5*] and high energy physics installations [*6*] [*7*]. Traditional architectures to build operating systems for those domains frequently match one of the following three categories: (i) state-based, (ii) rule-based or (iii) programmatic (flow-based).

Our analysis emerges from first hand experience evaluating a real system that has been put in place at CERN to operate one of the Large Hadron Collider experiments. As a result, we had to understand the limitations of traditional approaches.

In the search of an architecture, which better fits today's information systems, we follow the trail of web service oriented architectures that are tightly bound to workflows for coordination and control purposes [*8*].

Our experience report on real-scale, currently operating high-energy physics applications may help identifying key factors for modeling approaches that fit the requirements of distributed reactive systems .

II. Motivation

Having reviewed recently commissioned distributed data acquisition systems at CERN [*9*] [*10*] [*11*] [*12*] [*13*] we identified several limitations in traditional approaches to build systems for configuration, control and monitoring. In addition we found our results to be applicable to a much larger family of systems that we generally call *distributed reactive systems*. The motivation for this paper is to understand better the efficacy of different modeling approaches for generalized, distributed, reactive systems. Reactive systems, in contrast to transformational systems, usually terminate only on failure and generate responses to external stimuli as and when they occur [*2*]. The following section presents the data acquisition system of a project in which we are currently involved and that serves us as an example. The characteristics of the system at hand are not unique and can be found in systems akin as well as related application domains.

A. High Energy Physics

High-energy physics belongs to the domain of particle physics. Charged, subatomic particles are accelerated and brought to collision at high rate in intersecting storage rings. When particles interact at predetermined collision points in the ring, the total energy of the interaction partners is converted into matter. So new particles are created that escape in all directions. Every such interaction, called an event, is uniquely identified for later analysis. Multiple detector layers surround each interaction points (see Fig. 1). Each one is sensitive to a certain type of particles and hence newly created matter can be traced, quantified and identified. Groups of detector elements are connected to custom electronic readout devices that forward acquired data to deep buffers at high speed. Data from one detector collected in a single buffer are called a fragment. The sum of all fragments makes up the data of one event, which we can analyze using physics algorithms. For processing, a full event must be built in a single place. To cope with the high frequency at which event data are produced each event is built and analyzed in a separate processing unit of a computing cluster. This is possible, because collision events are independent from each other. Analysis of event data is based on a set of selection/rejection algorithms that are applied to the data in several steps. If an event does not satisfy the required properties, it is discarded, i.e. it will not be stored persistently for finer grained investigation.

Such computing systems are rather data bound than CPU bound. Data rates that we find in current systems are enormous. The Large Hadron Collider experiments having entered operation in September 2008 must eventually cope with an interaction rate of 40 MHz. Since no purely software-based distributed system may digest the total detector data of 1 MByte for a single event every 25 ns to date, pre-selection is performed in custom built, pipelined processors that reside close to the detectors. A resulting 100 kHz data rate to the fragment buffers poses still a challenging task. Each fragment is on average 2 KBytes large and about 500 buffers are needed to temporarily store the data. Buffers are essentially embedded computing systems equipped with IO processors, since the data have to be transferred over switched networks to the processors that host the analysis algorithms. Such a switched network requires an aggregate throughput of 800 Gbps with a total number of about 1000 logical ports. To perform the processing in time, i.e. before a fragment buffer overruns, a total of 5 TOps processing power is needed today. Since scaling of such a fully connected network has to be ensured, thorough research has been invested in efficient traffic shaping algorithms [*14*] [*15*] [*16*].

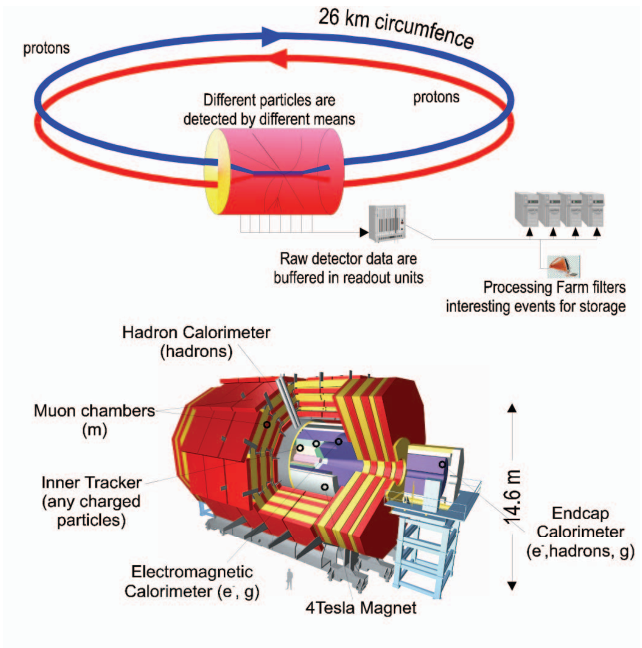
26 km circumfence
protons
protons
Different particles are detected by different means
Raw detector data are buffered in readout units
Processing Farm filters interesting events for storage
Hadron Calorimeter (hadrons)
Muon chambers (m)
Inner Tracker (any charged particles)
Electromagnetic Calorimeter (e⁻, g)
4Tesla Magnet
14.6 m
Endcap Calorimeter (e⁻,hadrons, g)

Fig. 1. The Compact Muon Solenoid experiment for the Large Hadron Collider at CERN.

B. Data Acquisition

A data acquisition (DAQ) system for an LHC scale high-energy physics experiment can be architected as a distributed computing system (see Fig. 2). In a DAQ cluster, a set of application modules, distributed over the nodes in the system, collaborate to perform data taking tasks. In contrast to number-crunching systems that dominate the parallel-computing domain, distributed systems for data acquisition are dominated by throughput and scaling requirements [*6*] [*17*]. The DAQ system is embedded in the experiment and interacts with custom hardware, detector readout and trigger system.

C. Case Study

The presented use case has been picked among a variety of scenarios. It is not limited to the system at hand but largely applies to other distributed reactive systems. We consider it a good use case for evaluation as it contains aspects that we require to find fulfilled by a modeling approach that targets distributed reactive systems. In particular we can analyze on how to model the following aspects with different modeling approaches:

- Synchronization in regards to dependencies between sub processes,
- Parallelization to efficiently model independent sub processes,
- Parameterization of models to deal with changes to the system layout transparently without changes to models.

In addition this use case allows us to analyze which protocols are natively supported by standard tools and if and how they deal with non-standard communication protocols.

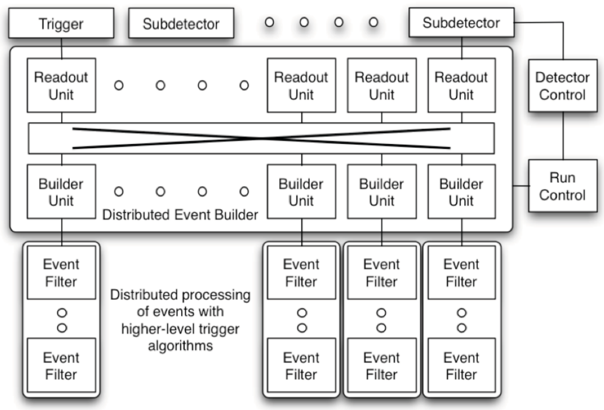

Fig. 2. An example for a distributed data acquisition architecture for a high energy physics installations.

Consider a *bootstrapping scenario* in which the operator brings the computers, networks and applications into a state ready for operation (Fig. 3). Initially all equipment is power cycled and the health status of the system is determined. The overall state is derived from numerous diverse constituents. Hardware gauges such as voltage levels and cooling need to be taken into consideration, as well as assuring proper functioning of system software such as drivers and daemons.

While some of the operations can be performed in parallel, others require respecting interdependencies among different components. Bootstrapping may continue for instance only if a connection test has completed successfully. At software level, directory servers must be operational before clients can run their local daemons that connect to it. Using service discovery with the service location protocol (SLP) [*18*] [*19*] requires that a directory agent is running before all other daemon processes on the participating computers. Reachability of computers is a prerequisite that is ensured by a properly configured DHCP environment, working name resolution in all subnets and a running SSH program on each node.

In reality, the system at hand is not limited to the central data acquisition partition, but is made up from a number of sub-systems, each one associated to a sub-detector that together make up the experiment. The outlined task serves therefore as a template that must be applied to all subsystems. This requires parameterization of the task with system and network configuration as well as with run-time parameters that are characteristic to each sub-system.

After presenting the model evaluation criteria, we match the case study to (i) state machines, (ii) statecharts, (iii) ECA rules and (iv) Web Workflows. State machines and statecharts were chosen over petri nets and other state-based modeling approaches as they are frequently used for scalable, distributed reactive systems found in high-energy physics experiments [*7*].

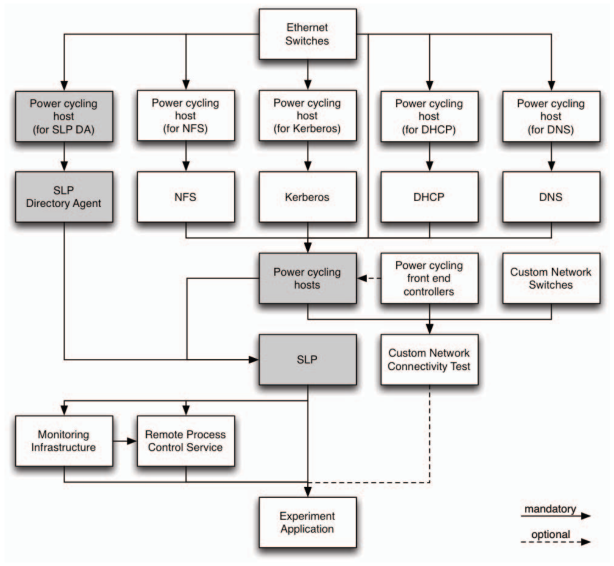
Ethernet Switches
Power cycling host (for SLP DA)
Power cycling host (for NFS)
Power cycling host (for Kerberos)
Power cycling host (for DHCP)
Power cycling host (for DNS)
SLP Directory Agent
NFS
Kerberos
DHCP
DNS
Power cycling hosts
Power cycling front end controllers
Custom Network Switches
SLP
Custom Network Connectivity Test
Monitoring Infrastructure
Remote Process Control Service
Experiment Application
mandatory
optional

Fig. 3. A simplified example for dependencies in a bootstrapping scenario of a data acquisition architecture in high energy physics installations. Elements with gray background are primarily used for analysis.

ECA rules were selected as a concrete example for a rule-based modeling approach that is used as core functionality in distributed multi-agent systems [*20*]. Finally the paper shows how the different models fit the evaluation criteria.

III. Model Evaluation

A. Evaluation Criteria

We base our evaluation criteria on perspectives as introduced in [*1*] [*21*] [*22*] [*23*]: (i) functional, (ii) organizational, (iii) informational, (iv) behavioral and (v) operational. Where required we add domain-relevant items (see TABLE I.).

At all steps of the evaluation our motivation is to find a small gap between specification and executed processes. Blending programming in the small to circumvent modeling deficiencies with high level modeling is however discouraged, since it introduces elements into models, which are unrelated to the processes and may directly influence its behavior [*1*] [*24*].

In addition to the standard perspectives, we have identified evaluation properties depicted in TABLE II.

We base our report on first hand experience that we gained by modeling various scenarios for an LHC experiment that is in operation now. From this we derive qualitative criteria.

TABLE I. Evaluation Criteria

Criteria	Description
Primary Criteria	
Functional	What is to be performed and why
Task	Capability to group behaviour
Task Decomposition	Organizing tasks into subtasks
Input	Modelling of unstructured input data
Output	Modelling of unstructured output data
Organizational	Who performs
Resources	Local or distributed resources
Participants	Programs, Humans
Structures	Grouping of resources
Roles	Roles can be assigned to participants
Informational	What information is required
Application data	Data manipulated by the process
Meta data	Associated with process data interpreted by model engine
Control data	All execution related data
Behavioural	When and how is it performed
Sequence	Dependencies
Parallel	Modelling of parallel processes
Constraint	Pre and post conditions
Temporal Constraint	Duration, timeout, real-time behaviour
Branching	(Conditional) splitting into multiple parallel processes
Loop	Repeat until a condition is true
Synchronization	Joins, rendezvous
Triggers	(External) events
Operational	How is it performed (interfaces)
Secondary Criteria	
Security	Who is allowed to access
Causality	To enforce consistency
History	What happened during execution
Integrity	Recovery from failures

TABLE II. ADDITIONAL EVALUATION PROPERTIES

Criteria	Description
Verification	Model in conformance with its goals
Syntax	Model in conformance with grammar
Semantic	Model in conformance with (process) goals
Structure	Model will not lead to erroneous execution

B. Finite State Machines

Finite state machines (FSM) primarily model behavior based on states and transitions. A state is the sum of all past events, and transitions model changes of states on events (triggers) [*5*]. Flat FSMs only allow one state to be active at any point in time. They are not capable to directly model aspects found in our use case: (i) functional, (ii) behavioral (parallelism, synchronization) and (iii) organizational.

Functional, task decomposition: Exclusively modeling behavior and not being able to model task decomposition is an exclusion criteria for our application domain. In our case study either we would have to model all actions in a single, flat state machine or provide one state machine for each task to be performed. The problem of the latter approach is that the model does not allow us to bond together individual state machines. As a matter of fact various products extend FSMs with this feature at implementation level [*7*].

Behavior, parallelism: Our scenario contains independent tasks that can be performed in parallel, like checking SLP and SSH server on each node. Flat FSMs do not allow to directly model this behavior. Representing all tasks in a single state machine leads to state explosion (Fig. 5) [*25*].

Behavior, synchronization: As an example for a double dependency of a service, consider the SLP daemon on a node that requires a central directory agent to be operational and a valid configuration file. The events to acknowledge the presence of either precondition can be received in any order. Therefore, modeling this task as a strictly sequential procedure to overcome the lack of modeling capabilities becomes difficult. Considering all combinations of event occurrences makes the state machine grow exponentially (Fig. 4).

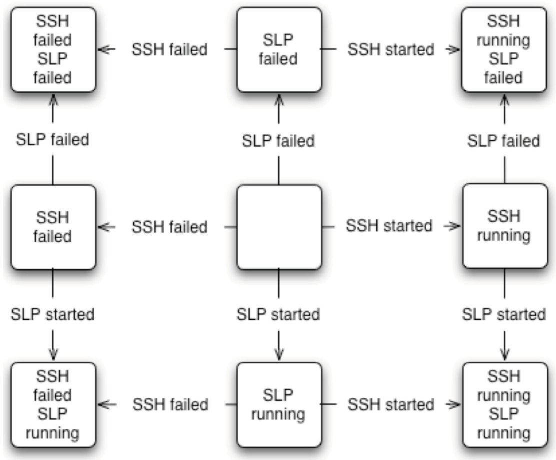

Fig. 5. Checking of two services (SSH and SLP) in parallel with flat state machines.

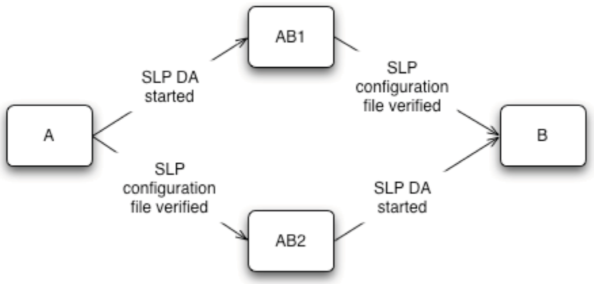
AB1
SLP DA started
SLP configuration file verified
A
B
SLP configuration file verified
SLP DA started
AB2

Fig. 4. Modelling a point of synchronization with two unordered incoming events.

Organizational: Our system is composed of several subsystems, built from a single blueprint. Each subsystem maps to a set of network addresses, subnets, racks, power distribution units and data acquisition circuits. An adequate model must be able to separate the generic task description from run-time data needed to achieve its goals. This separation is not foreseen in FSMs.

C. Statecharts

Statecharts as introduced in [*25*] aim at overcoming shortcomings of flat FSMs by providing a collection of extensions: (i) clustering and refinement of states, (ii) orthogonality and (iii) generalization of transitions. Those concepts provide functional task decomposition and behavioral modeling for parallel tasks including splits and joins as well as input, output and temporary constraints. Some organizational and operational deficiencies remain.

Organizational: The capability to support timeouts is paramount to model distributed reactive systems. Although the blueprint includes timeouts for the self-tests of the data-acquisition hardware connected to particle detectors, these timeouts vary with the subsystems depending on type of hardware and number of cards that need to be tested. Statecharts include a notation for specifying timeouts. Timeouts trigger the emission of events. Although it is possible to model timeouts with parameters there is no means to bind these parameters to a subsystem definition that is part of the organizational perspective (Fig. 6).

Operational: Staying with the above outlined example, we see that our system is not closed but needs to interact with other systems, when testing detector hardware electronics. Integration requires interface specifications at model level. Irrespective of specifying a test as an invocation of an external program or as interactions with hardware registers at low level,

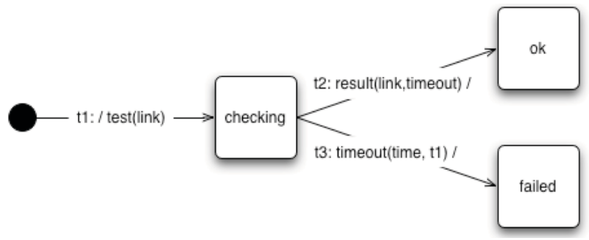

Fig. 6. Statecharts with scripts (t1), parameterized events (t2) and parameterized timeouts (t3).

input parameters and results must be defined at the model level. Since statecharts do not include the notation of modeling resources and other organizational components, any action specification with parameters remains colloquial and is not verifiable against the specification.

Statecharts became standardized in the context of UML and are thus a viable solution to model parts of distributed reactive system [*26*]. However, they need to be combined with other UML diagrams, such as interaction, timing and structure diagrams, to model a system completely. UML does, however, not specify how the diagrams are integrated.

D. Event Condition Action Rules

Event Condition Action (ECA) rules autonomously react to an *event* by performing *actions* when the *condition* evaluates to true [*27*] (Fig. 7).

```
on event if condition do actions
```

Fig. 7. Standard form of ECA rules.

ECA rules are inherently parallel where an event may have attached additional information. Actions are scripts that change variables or emit new events. The lack of grouping functionality and association between rules, leads to problems for modeling (i) functional, (ii) organizational and (iii) operational perspectives.

Functional, task decomposition: The lack of any support to structure the model imposes limitations on the size of the system that can be described.

Organizational: Since no structuring is foreseen, organizational components cannot be modeled explicitly. They exist however implicitly by the presence of parameterized events.

Behavioral, synchronization: Considering a dependency on two events that may occur in any order, we unveil another limitation of the model. We take again the example of starting the SLP daemon on each system node that depends on a reachable directory agent and a valid local configuration file. This case can only be modeled indirectly though the use of conditions and assuming that variables exist (Fig. 8). This complication could be avoided by enforcing sequential execution of checking configuration file validity and checking for a running directory agent. This introduces additional knowledge about internals of other subsystems, leading to error prone system design. [*28*] presents a mapping of behavioral aspects to ECA rules using patterns.

```
on init
        if exists(/etc/slp.conf)
        do SLPCONF=true; emit(startslp)
on started(SLPDA)
        do SLPDA=true; emit(startslp)
on startslp
        if SLPDA==true AND SLPCONF==true
        do action()
```

Fig. 8. Modelling synchronization point with ECA rules.

Operational: In the above outlined example, rules operate on diverse subsystems. There is no formal description available (a) to enforce the validity of interfaces and (b) to model encapsulation.

Although ECA rules represent only a partial view of a system model, they became popular in distributed reactive systems with the advent of tool support to counterfeit the problems induced by the lack to structure rules. As soon as rules can be set into context of objects to which they apply, the technology allows to come to scalable models in a scalable manner, since all points listed under functional and organizational aspects can be realized. Initially tool support was highly custom, e.g. through the introduction of data points [*29*]. Through the introduction of the IEC 61499 standard [*30*], ECA rules have evolved into a model that fits better the real world. Most important, the problem of scheduling rule execution has been addressed by introducing the concept of Execution Control Charts (ECC). Rules are associated to states of the functional block. What remains to be addressed in this extended model is the organizational domain. Currently it is still difficult to produce abstract tasks that afterwards can be mapped to concrete system configurations. The model is very much tied to industrial control systems, but has great potential to evolve into a generalized methodology.

Research is still ongoing for standardizing the execution model that is a prerequisite to validate that a model exhibits the same run-time behavior in different scenarios using different execution platforms. The approach taken with ViDRE to execute business rules was to define a standardized Web Service interface that can be implemented with different rule engines [*31*] [*32*] [*33*]. For distributed reactive systems as outlined in this paper, we however need the model to interact directly with the distributed reactive system. In particular the model needs to be able to communicate with application services to evaluate conditions of and to perform actions on application services.

E. Workflows

We see the Business Process Execution Language (BPEL) as the unified approach for modeling Web Workflows. It is based on individual approaches by IBM, Microsoft and Oracle. Workflows are a formal representation of a process to model reactive systems. They have their roots in modeling document-based executable business processes. The separation of Web Services from their underlying technologies made it possible to define a technology independent programming model. The choice for this model in the service-oriented architecture (SOA) fell on workflows. While Web Service standards are concerned with describing service interfaces, Web Workflows aim at orchestrating these services. Actually, what is denoted by *Web Workflows* is entirely decoupled from concrete implementations, systems and technologies and represents the distilled essence as outlined in XML, WSDL and BPEL. This concept is effective because it abstracts from real-world scenarios as opposed to traditional modeling concepts that must be mapped to existing technologies [*34*].

We extend the example of starting an SLP daemon, which depends on a reachable directory agent and a valid configuration by modeling the checking the availability of a central directory agent as a separate workflow with configurable timing constraints (Fig. 9). Based on this, we analyze the different perspectives.

Behavioral: Workflows natively support parallel flows of activities, branching, synchronization and loops. Sequential flows have to be explicitly modeled using transitions between activities [*35*]. Activities with multiple outgoing connections represent unconditional branching, and activities with multiple incoming transitions are points of synchronization and allow specifying a join condition.

Functional: Workflows allow decomposing systems into tasks, which are either atomic activities (e.g. wait, receive, reply and assign) or workflows.

Operational: Workflows are Web Services with well-defined interfaces in WSDL that allow attaching parameters to requests and responses, and let us validate messages.

Informational: Application data are formulated in XML and transformations can be modeled with explicit assignment activities formulated as XPath expressions.

Organizational: Our system consists of several subsystems, built from the same blueprint. Although Workflows support generic modeling of tasks, and activities can be fully parameterized with variables, binding of organizational data to subsystem definitions is not foreseen. Instead the organizational aspects must be modeled with the help of other primary perspectives.

Integrity: Failures are modeled as exceptions. Thus, error recovery can be modeled in fault handlers (catch clause) close to the execution flow [*36*].

Although Web Workflows provide native support for four out of five primary perspectives, some scenarios found in our use case still cannot be modeled efficiently.

Verification: Verifying the model is not possible for all perspectives. The organizational perspective is modeled with the help of informational and operational aspects and can only be checked at runtime. Simulating the system also proves to be difficult, as Workflows interact with unpredictable real-world systems.

Behavioral, dynamic parallel flows: Starting all SLP daemons requires that all computers that host SLP daemons are booted and that the directory agent is reachable. As the number of hosts is configuration dependent, we only model the sequence of booting a single host. Starting all the SLP daemons is performed in a parallel loop. This, however, prevents us from efficiently modeling the second dependency from outside the loop (Fig. 10). Either all hosts have to be booted and a central SLP directory agent is running before the first SLP daemon is started (Fig. 10a) or a central SLP directory agent is running before any SLP directory agent is started (Fig. 10b). Either approach introduces a bottleneck.

Web Workflows provide a complete model that tightly integrates all five perspectives. They can be directly executed and allow integration with existing services through standard Web Service interfaces.

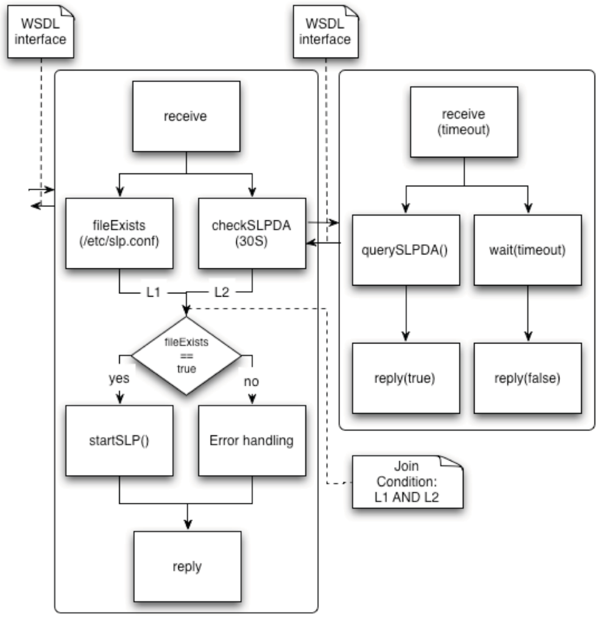

Fig. 9. Modelling SLP startup with Web workflows.

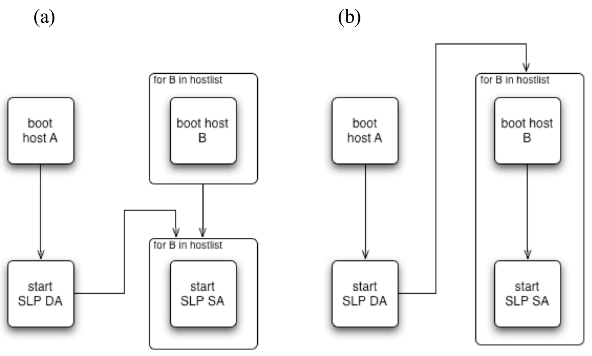

Fig. 10. SLP daemons depend on the local host to be booted and a central SLP directory agent.

IV. Conclusion

Our primary motivation is to keep the gap between specification and executed processes small. To keep additional complexity induced by working around deficiencies in model features small, we foster a modeling approach that natively supports all outlined perspectives. TABLE III. gives an overview of how well the different models support the evaluation criteria based on our experience that we gained in modeling scenarios in our project: (+) natively supported, (o) supported with specific patterns and (-) not foreseen or significant complexity increase.

Traditional modeling concepts have their roots in designing monolithic and often in-silicon systems. These models do not easily fit today's requirements on distribution, system size, extensibility and a growing degree of complexity as well as dynamic reconfiguration. Other perspectives, such as security are entirely unforeseen and therefore almost unthinkable to

TABLE III. EVALUATION RESULTS

	Finite state machines	Statecharts	ECA rules	Web Workflow
Functional	-	+	-	+
Organizational	-	o	-	o
Informational	-	-	+	+
Behavioural	-	+	o	+
Operational	-	-	-	+
Security	-	-	-	-
Causality	+	+	-	-
History	+	+	-	+
Integrity	-	-	+	+
Verification	+	o	o	o
Gap	large	medium	medium	small

retrofit. IEC61499 is a valid candidate to unify statecharts and ECA rules into a new, complete modeling architecture that has enough support from industry and academia to penetrate many domains. On the other side the standard originates from industrial control and has therefore a strong focus on modeling closed and embedded systems.

Web Workflows on the other side emerged as an abstraction from real-world application scenarios and are still evolving with them. Models are strongly concerned with interfaces and orchestrating services described by those interfaces. Here, the problem lies rather in fitting the execution support to the tight constraints of high-performance distributed real-time systems, rather than fixing modeling deficiencies.

As shown in this qualitative evaluation, finite state machine approaches as prevailing in physics experiments have a significant gap between model and execution. This lead to development of custom tools that bridge this gap with custom extensions [*7*]. Web workflows on the other hand support four out of the five views natively and one organization view using the informational view. They show a small gap between specification and executed processes and allow direct execution of models assuming that communication with the services is based on SOAP. Web Workflow tools such as ActiveBPEL also allow to extend the workflow engine to support additional communication protocols that can be used by models transparently through custom invocation handlers [*37*].

As a result of our comparison work, we chose to launch a project to evaluate Web Workflows for modeling, designing and executing the scenarios outlined in this article. Initial results of the ongoing project are encouraging, but if workflows are indeed a novel technology that can address the described kind of systems still remains to be elucidated in detail.

ACKNOWLEDGMENT

We wish to thank the CMS collaboration for providing the infrastructure to carry out the project.

REFERENCES

[1] B. Curtis, Kellner M., and J. Over, "Process Modeling," *Communications of the ACM*, vol. 35, no. 9, pp. 75-90, 1992.

[2] D. Harel and A. Pnueli, "On the Development of Reactive Systems," *Logics and models of concurrent systems*, pp. 477-498, 1985.

[3] J. Gutleber, G. Schimak, and H. Humer, "Using Active Behaviour in Environmental Monitoring Systems," in *IFIP TC5 WG5.11 International Symposium on Environmental Software Systems*, 1997, pp. 128-135.

[4] H. Gjermundrod, D.E. Bakken, C. H. Hauser, and A. Bose, "GridStat: A Flexible QoS-Managed Data Dissemination Framework for the Power Grid," *IEEE Transactions on Power Delivery*, 2008.

[5] Y.R. Martin, S. Coda, B.P. Duval, X. Llobet, and J.-M. Moret, "A new plant control software for the TCV tokamak," in *paper presented at ICALEPCS 2005*, 2005.

[6] J. Gutleber, R. Moser, and L. Orsini, "Data Acquisition in High-Energy Physics," in *Astronomical Data Analysis Software & Systems XVII*, 2008, pp. 47-56.

[7] B. Franek and C. Gaspar, "SMI++ object oriented framework used for automation and error recovery in the LHC experiments," *J. Phys.: Conf. Ser.*, vol. 219, no. 2, 2010.

[8] F. Daniel and B. Pernici, "Web Service Orchestration and Choreography: Enabling Business Processes on the Web," *E-Business Models, Services, and Communications - Advances in E-Business Research Series*, vol. 2, pp. 251-274, Nov. 2007.

[9] The ALICE Collaboration, "The ALICE experiment at the CERN LHC," *JINST 3 S08002*, 2008.

[10] The ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider," *JINST 3 S08003*, 2008.

[11] The CMS Collaboration, "The CMS experiment at the CERN LHC," *JINST 3 S08004*, 2008.

[12] The LHCb Collaboration, "The LHCb Detector at the LHC," *JINST 3 S08005*, 2008.

[13] L. Evans and P. Bryant, "LHC Machine," *JINST 3 S08001*, 2008.

[14] E. Barsotti, A. Booth, and M. Bowden, "Effects of various event building techniques on data acquisition system architectures," in *Fermilab note FERMILAB-CONF-90/61*, Batavia, IL, USA, 1990.

[15] G. Bauer et al., "Effects of Adaptive Wormhole Routing in Event Builder Networks," *IEEE Transactions on Nuclear Science*, vol. 55, no. 1, pp. 182-189, Feb. 2008.

[16] G. Bauer et al., "The data-acquisition system of the CMS experiment at the LHC," *J. Phys.: Conf. Ser.*, vol. 331, no. 022021, 2011.

[17] J. Gutleber, S. Murray, and L. Orsini, "Towards a homogeneous architecture for high-energy physics data acquisition systems," in *Comp. Phys. Comm.*, 2003, pp. 153:155-163.

[18] E. Guttman, C. Perkins, J. Vaizades, and M. Day. (1999) Service Location Protocol, Version 2. [Online]. HYPERLINK "http://www.ietf.org/rfc/rfc2608.txt" http://www.ietf.org/rfc/rfc2608.txt

[19] E. Guttman. (2002, Jan.) Vendor Extensions for Service Location Protocol, Version 2. [Online]. HYPERLINK "http://tools.ietf.org/html/rfc3224" http://tools.ietf.org/html/rfc3224

[20] K. Taveter and G. Wagner, "Agent-Oriented Enterprise Modeling Based on Business Rules," in *ER '01 Proceedings of the 20th International Conference on Conceptual Modeling: Conceptual Modeling*, 2001, pp. 527-540.

[21] S. Jablonski and C. Bussler, *Workflow Management: Modeling Concepts, Architecture, and Implementation.*: International Thomson, 1996.

[22] M.M. Kwan and P.R. Balasubramanian, "Dynamic Workflow Management: A Framework for Modeling Workflows," in *Proc. 13th Hawaii International Conference on System Sciences*, vol. 4, 1997, pp. 367-376.

[23] M. Zur Muehlen, *Workflow-based Process Controlling: Foundation, Design, and Application of Workflow-driven Process Information Systems.*: Logos, 2004.

[24] F. DeRemer and H. Kron, "Programming-in-the-large versus Programming-in-the-small," in *Proc. int. conference on Reliable software*, 1975, pp. 114-221.

[25] D. Harel, *Statecharts: A visual formalism for complex systems*. Rehovot, Israel: Department of Applied Mathematics, The Weizmann Institute of Science, 1987.

[26] S.W. Ambler, *The Object Primer. Agile Model-Driven Development with UML 2.0.*: Cambridge University Press, 2004.

[27] J. Iturrioz, O. Díaz, and I. Azpeitia, "Reactive tags: associating behaviour to prescriptive tags," in *HT '11 Proceedings of the 22nd ACM conference on Hypertext and hypermedia*, New York, 2011, pp. 191-200.

[28] L. Chen, M. Li, J. Cao, and Y. Wang, "An ECA Rule-based Workflow Design Tool for Shanghai Grid," in *2005 IEEE International Conference on Services Computing*, vol. 1, 2005, pp. 325-328.

[29] S. Schmeling, "Common tools for large experiment controls: A common approach for deployment, maintenance, and support," *IEEE Transactions on Nuclear Science*, vol. 53, no. 3:1, pp. 970-973, 2006.

[30] G. Cengic, O. Ljungkrantzand, and K. Akesson, "A Framework for Component Based Distributed Control Software Development Using IEC 61499," in *IEEE Conference on Emerging Technologies and Factory Automation*, 2006, pp. 782-789.

[31] C. Nagl, F. Rosenberg, and S. Dustdar, "ViDRE - A Distributed Service-Oriented Business Rule Engine based on RuleML," in *10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06), 16. - 20. October 2006*, 2006.

[32] F. Rosenberg, C. Nagl, and S. Dustdar, "Applying Distributed Business Rules - The ViDRE Approach," in *IEEE International Conference on Services Computing (SCC'06), Services Computing Contest (SOA Contest 2006), 18. - 22. September 2006*, Chicago, USA, 2006.

[33] F. Rosenberg and S. Dustdar, "Business Rules Integration in BPEL - A Service-Oriented Approach," in *7th International IEEE Conference on E-Commerce Technology (CEC 2005), 19 - 22 July 2005*, Munich, Germany, 2005.

[34] M. Vasko and S. Dustdar, "A view based analysis of workflow modeling languages," in *IEEE PDP 2006 - 14th EUROMICRO Conference on Parallel, Distributed and Network-based Processing, 15 - 17 February, 2006*, Montbéliard-Sochaux, France, 2006.

[35] N. Russell, A. ter Hofstede, W. van der Aalst, and Nataliya Mulyar, "Workflow ControlFlow Patterns: A Revised View," , 2006, http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf.

[36] R. Shapiro, *A Technical Comparison of XPDL, BPML and BPEL4WS.*: Cape Visions, 2002.

[37] T. Dornemann, E. Juhnke, and B. Freisleben, "On-Demand Resource Provisioning for BPEL Workflows Using Amazon's Elastic Compute Cloud," in *CCGRID '09 Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 140-147.

[38] F. Daniel and B. Pernici, "Insights into Web Service Orchestration and Choreography," *International Journal of E-Business Research*, vol. 2, no. 1, pp. 58-77, 01-03 2006.