TOWARDS MODEL-DRIVEN DESIGN OF SUBSTATION AUTOMATION SYSTEMS

Rogério PAULO*, Adriano CARVALHO†
* EFACEC Sistemas de Electrónica, S.A. - Portugal
rdpaulo@se.efacec.pt
† Faculdade de Engenharia da Universidade do Porto - Portugal
asc@fe.up.pt

INTRODUCTION

This paper presents a meta-modelling architecture suitable for model-driven design of substation automation systems (SAS). The model-driven approach applied to SAS design is presented, the requirements for an integrated engineering environment are pinpointed and a modelling language is proposed as an approach towards the development of SAS design and integration enabling tools.

CURRENT TRENDS

Substation automation, as well as other utility automation domains, requires adequate design tools in order to cope with increasingly demanding budget, resource and time-to-deployment requirements in an engineering domain where information complexity and size are growing [1]. In the light of recent standardization, namely IEC 61850 [2], two development trends are established: a shift from data point based modelling towards object-orientation and a shift from document-driven design towards model-driven design. These trends mark the next generation of SAS design, which can be achieved by handling system object models in an integrated engineering environment.

Automation Platforms

Automation systems are evolving towards ubiquitous, diverse, broad, agile, autonomous and integrated large-scale real-time information systems [3]. Information handling has become a crucial issue, given the need for cost-effective integration of multiple application domains such as monitoring, supervision, control, protection, asset management, maintenance, condition monitoring, incident management, power quality monitoring, configuration management or security management. With the purpose of optimizing information handling, object orientation is being widely introduced at the platform level.

System Development

The SAS development process follows a multi-stage cascaded process from requirements elicitation and analysis to factory and site testing. In this process the specification and design stages are the most value-adding activities. However, in today’s industrial practice, the activities performed in these stages lack appropriate tool and methodological support. IEC 61850 is a major step forward towards SAS model-driven development, however, user interface and behavioural modelling, as well as overall modelling language support are lacking.

AUTOMATION SYSTEMS DESIGN

Model-driven Design

The application of the Model-driven Architecture [4] (MDA) to SAS development is aimed at integrated system-oriented design. Model-driven design involves designing and validating reusable system models which are then transformed into platform-specific models, deployable to the secondary system (fig. 1).

The system models include platform-independent models of the SAS, which comprise logical functions, including function interfaces, internal structures and algorithms, as well as user interface components. These models can be designed with SAS-oriented tools based on an extension of the domain logical interface models proposed in IEC 61850 to include behaviour and user interface. The platform-independent models are highly reusable and can be mapped to platform-dependent models that match a given system architecture. Early validation of these system models is feasible, allowing early error correction. The set of device-independent and device-dependent models is then automatically transformed into device-specific models (configuration parameters) and deployed into the physical system. The model-driven design approach can be applied throughout the SAS development process improving reuse, facilitating communication between engineers and data exchange between tools, automating low value-adding activities and reducing rework.

Requirements

The application of the model-driven approach presents three requirements: (i) the acceptance of domain models in the industrial community, (ii) the availability of system-oriented development tools and (iii) the establishment of an applicable
The Unified Modelling Language (UML) is the reference development of cost-effective flexible engineering tools. Not only a requirement for effective design, but also for describing both domain models and system models. This is availability of adequate modelling languages capable of supporting domain model interfaces and model interchange formats. These initiatives are contributing to world-wide acceptance of domain models.

Engineering tools. The availability of computer-aided engineering tools for supporting modelling, model validation and automatic deployment are a fundamental issue for taking the model-driven approach to practice. Given near-term availability of accepted domain models, building modelling tools is a software development issue. The automatic mapping of logical system models to device dependent parameter sets in a distributed multi-vendor and mixed-generational environment is, however, not yet straightforward, even in the light of standard configuration languages or standard programming languages. Substation automation today is device integrated and cooperation between IEDs of different vendors is possible due to the generalized adoption of open communication protocols. Inter-vendor engineering integration is, however, rather inexisten.

Modelling languages. The third requirement is the availability of adequate modelling languages capable of describing both domain models and system models. This is not only a requirement for effective design, but also for the development of cost-effective flexible engineering tools. The Unified Modelling Language (UML) is the reference language, however, its generic application scope and feature-rich model renders UML inadequate for application in automation systems design. Moreover, the integration of current programming or modelling languages (IEC 61131-3, IEC 61499, etc.) with UML and the IEC 61850 models is not straightforward, hindering effective support for integrated design of automation systems.

INTEGRATED DESIGN ENVIRONMENT

Requirements

An integrated model-driven design tool environment shall support (i) platform-independent system modelling, (ii) platform-dependent design and mapping, and, (iii) behavioural and static consistency validation. Platform-independent design involves (i) establishing SAS-oriented models of the primary system, (ii) specifying and designing the SAS functions, including distributiveness and interfaces, as well as internal structures and algorithms, and, (iii) user interface design. The platform-dependent design targets (i) physical architectural design, including devices and communication networks, (ii) process interface definition and (iii) device parameter generation. Other project support functions such as documentation generation, version management, and model storage and retrieval shall also be available.

Reference Architecture and Interfaces

A reference tool environment, its inputs and outputs, based on IEC 61850-4 [6] is presented in fig. 2.

The system tool encapsulates all functionality from the automation engineer perspective and would be used for the design, simulation, validation and transformation of SAS models, as well as real-time monitoring of systems for testing. Design is performed by assembling partial existing models, obtained from a model database or from external sources, and models designed within the tool environment. Device tools, seamlessly integrated in the environment, would interface actual devices for configuration, operational parameter extraction and setting, as well as real-time device monitoring and operation. The documentation tool produces human-readable documents from SAS models.

THE META-MODELLING ARCHITECTURE

Overview

As a contribution towards the adoption of SAS model-driven design a modelling architecture based on the OMG four-layered meta-modelling infrastructure is presented (fig. 3).

The core of this architecture is a modelling language named
Automation Systems Modelling Language (ASML), a MOF\cite{7} meta-meta-model subset instance. Automation system models are instances of the ASML, including both domain models such as IEC 61850 and engineered system models. Lower level models correspond to internal device objects running in a specific physical system.

**Modelling Scope**

The SAS model artefacts (fig. 4) that the language must handle draw from the presented requirements for the design environment and can be described by three distinct interrelated components \cite{2}: (i) static structure: structured simple typed data elements and complex relationships, (ii) behaviour: discrete event-driven, with real-time requirements and exhibiting concurrency and (iii) graphical representation and user interaction: graphical representations for system documentation, design, testing and operation, defined as interactive, animated, vector-based bi-dimensional graphics.

**Modelling Language**

The approach towards the ASML specification was the extension of a sub-set of UML\cite{5}. The core model is based on a simplification of the UML static structure model, in which the generalizable classifier (fig. 5) is the integrating element, encapsulating data, behaviour and user interaction. Classes are used for modelling all system artefacts such as primary equipment, IEC 61850 logical nodes and data, devices, communication profiles, etc. Artefact associations are modelled via class relationships such as associations and compositions. Objects are created and linked to build each specific system from existing model blocks, defined by classes.

A class definition includes generalizable features such as classified attributes and operations, but also generalizable interaction and behavioural elements. Behavioural elements specify object discrete event dynamics, for which hierarchical interpreted timed Petri nets are proposed. The interaction element meta-model is based on the SVG\cite{8} declarative, animated and interactive graphics model, incorporating connectivity. The interaction elements are primarily aimed at synoptic design and alternate model graphical representation. The conditions, actions and procedures associated with functions, behavioural elements and interaction elements are described according to a meta-model adapted from IEC 61131-3\cite{9} Structured Text or Function Block Diagram.

To allow the application of domain models such as IEC 61850, the UML extension mechanisms (constraints, tagged values and stereotypes) are also included in the ASML.

---

**Fig. 4. A model of SAS model artefacts.**

**Fig. 7. ASML meta-model.**
**Tool Architectural Design**

The presented architecture facilitates the implementation of the design tool core modules such as the model repository, the user interface or device specific compilers. An overview of the proposed design tool architecture and its dependencies on the meta-modelling architecture is shown in fig.6.

Fig. 6. The meta-modelling architecture and tool dependencies.

**APPLICATION ILLUSTRATION**

**Design Process**

The core of the SAS design process (fig. 7) does not change by using either object-orientation or model-driven design. It is driven by both the requirements and the technological feasibility of the solution [10].

In the specification stage an incomplete ASML model is produced. This specification model includes a detailed primary system model, incomplete logical and secondary system models and annotations covering additional constraints and requirements. During the system design and validation stages a complete detailed model of the SAS is produced. System design is performed according to a mixed process in which both bottom-up or a top-down approaches are used. The bottom-up-based approach involves the aggregation of well known devices, building the system functional model from the secondary model. The top-down-based approach comprehends the establishment of validated platform-independent model which is then mapped to a secondary system model.

**SAS Domain Model**

A simplified example of an SAS domain model based on the IEC 61850 standard series is presented.

**Platform-independent system.** Platform-independent models are built from IEC 61850 logical nodes (LN), which are sub-functions. LNs are specified by stereotyped active classes including behaviour and user interaction (fig. 8). The interface of these classes is composed of standardized data and methods included in IEC 61850 data classes. The system is designed by specifying and instantiating logical nodes and their relationships.

Fig. 7. The design process.

Fig. 8. Platform-independent system modelling.

Fig. 9. Primary system modelling.

A substation is mainly composed of active devices such as breakers, passive devices like lines and bars and electrical...
connections between devices that are not monitored or controlled but establish substation topology. LNs may reference primary devices in order to model monitoring, control or virtualization.

**Secondary system.** Secondary system models include IED instances and their relationships with primary and logical models (fig. 10). An *IED* stereotyped class may include data to establish device-specific parameterization, but major IED functions are specified by associated LNs. Each IED may also be associated with communication systems modelled by classes stereotyped with server, client and access-point. These stereotypes classify classes that model logical and physical networks and include protocol specific parameterization.

CONCLUSIONS

A contribution towards model-driven design of substation automation systems was presented. A meta-modelling architecture and an object-oriented modelling language applicable to the description and validation of system models was proposed. Illustration of the modelling approach, applied to the design process, shows that the adoption of object oriented modelling and the model-driven design approach leads to both cost-reduced and quality-enhanced system development.

ACKNOWLEDGMENT

The authors gratefully acknowledge the contributions and support of António Carrapatoso, EFACEC Sistemas de Electrónica, S.A.

REFERENCES


