INTEGRATED MONITORING AND CONTROL SPECIFICATION ENVIRONMENT

C. Subhrojyoti, N. Swami, TRDDC, Pune, India
Harshal G. Hayatnagarkar, Pune, India

Abstract

Monitoring and control solutions for large one-off systems are typically built in silos using multiple tools and technologies. Functionality such as data processing logic, alarm handling, UIs, device drivers are implemented by manually writing configuration code in isolation and their cross dependencies maintained manually. The correctness of the created specification is checked using manually written test cases. Non-functional requirements – such as reliability, performance, availability, reusability and so on – are addressed in ad hoc manner. This hinders evolution of systems with long lifetimes. For ITER [1], we developed an integrated specifications environment and a set of tools to generate configurations for target execution platforms, along with required glue to realize the entire M&C solution. The SKA [2] is an opportunity to enhance this framework further to include checking for functional and engineering properties of the solution based on domain best practices. The framework includes three levels: domain-specific, problem-specific and target technology-specific. We discuss how this approach can address three major facets of complexity: scale, diversity and evolution.

INTRODUCTION

The typical life cycle phases of any software development project starts with analysis such as stake holder needs, system requirements, architecture choices followed by detailed design of the target system, its implementation, verification and validation, integration and testing. Execution of each of these life cycle phases is usually supported by a collection of tools and technologies which enable maintaining traceability across these phases. Doors[3], SysML[4] are example which provide support during requirements analysis, architecture and design of software systems. Although in theory it is possible to use SysML all the way up to generating the final executable code from the high level design model, in practice it is seen that it is not very strong in its adaptation to building Monitoring and Control (M&C) solutions for real-time systems. This is due to various reasons such as difference in paradigm between modeling and implementation domain, lack of support for project or domain specific needs and above all it is a lot of work to model all aspects of M&C using a modeling tool then generate code from it.

Our participation in the ITER project in the design and development of their M&C solution and later with SKA project for their M&C aspects helped us appreciate the need for building methodologies and tools that enhances the current state of the art to cater to the needs for building M&C solutions for projects like ITER and SKA. Our belief is that we can leverage from the existing tools and methodologies like SysML and our knowledge from the M&C and related domains, to create a complete specification model for M&C. This specification model can then be instantiated through a specification environment to build M&C solutions spanning across different physics domains. This paper presents the idea in the context of ITER and SKA. The first section of the paper describes the current state of practice highlighting the methodologies and tools typically used in a software development projects. Second section describes the approach taken by ITER to improve upon the existing practice. Third section describes how the lessons learnt from ITER could be enhanced further so that it can be useful for other projects like SKA.

STANDARD PRACTICE

The current trend to architecting and designing both software and non-software including M&C systems is to use techniques from the system engineering methodology. The entire process can be understood from the figure 1.
As per our experience, the identification of the design requirements, components along with their functionalities, engineering qualities, relationships and dependencies are ideally captured and analyzed in the first two steps in the life cycle. For example, the analysis of stakeholder objectives and system requirements typically get captured as texts supported by tools that allow them to be managed for version control and so on. Then the requirements are analyzed and elicited further through modeling and meta modeling tools such as SysML. These tools allow modeling the architecture and design of the system. Translating the design into actual realization happens in the third and the fourth step. These step is typically a manual activity where the design created using the modeling tool is typically translated manually to either to SCADA specific input format which internally translate them to lower level executable or directly into C/C++ code.

Challenges with Existing Practice

We find the approach of model based development useful for building M&C solutions but not without caveats. Some of the challenges faced while using the standard approach along with the associated tools and technologies are mentioned below:

- The notion of stakeholder objectives and values don't get properly translated into the design and hence loose tractability.

ISBN 978-3-95450-139-7

Copyright © 2014 CC-BY-3.0 and by the respective authors

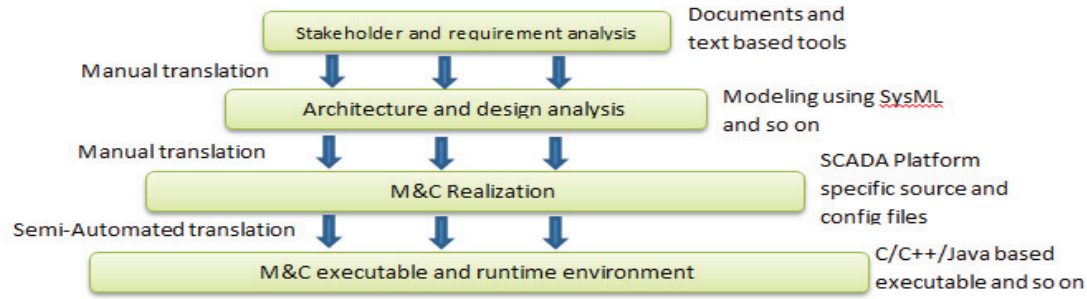

Figure 1: Typical life cycle of the M&C realization process.

- Capturing and analysis of the quality aspects in the design phase and making sure that they translate into the subsequently realization strategy is not very well supported in the system engineering modeling environment.
- Although support for code generation for standard programming languages such as C++/Java from SysML model exists, there is lack support for more specific platforms such as EPICS [5]. Support for round tripping to pull manual changes in the generated source code back into the model becomes hard due to difference in paradigm ,i.e. programming language vs configuration source files.
- Since the model is manually translated there is a huge possibility of having inconsistencies such as creating duplicate implementation of components in the realized code, inconsistency in names, ad hoc interfaces and so on.
- Integrating the individual realized pieces of the M&C solution developed across the geography into the large M&C environment becomes a challenge.
- Also the more generic system engineering tools are not very well aligned to the M&C domain for which it is being used and hence it is not possible to do domain specific analysis of the design created using these modeling environments.
- It is difficult for the generic system engineering tool to adapt to the domain or problem specific vocabulary.

ITER APPROACH

In order to mitigate some of the above challenges specially the ones related to geographically distributed development teams, ITER conceptualized the Self-Description Design (SDD) Editor Framework. This framework aims to incrementally integrate Instrumentation and Control (I&C) components built by various teams across geography for the ITER project. The idea is to provide a full integrated view of the systems and subsystems which are under the purview of CODAC, the supervisory M&C system at ITER. The SDD editor allows specification of the I&C components along with their structures, properties and functionalities using a common abstract vocabulary created by the ITER team which is then automatically translated into the underlying M&C implementation technology specific format which are EPICS, CSS and S7 PLC.

As part of this framework, a set of tools have been implemented based on the concept of model-driven development, which facilitate capturing high-level specification of ITER's I&C component into a domain model, checking for various types of inconsistencies in the model, generating code specific to target technology such as EPICS and CSS and also, in some cases, retrofitting the changes made in the generated code back into the specification model.

SDD Editor Components

The architecture components of the SDD editor are briefly explained below -

Model: Central to this environment lies ITER's M&C model, which acts as mould to hold high-level specification of the M&C system description called as 'Self-description data' (SDD). In the philosophy of model-driven development, this model acts as 'Platform-independent model' (PIM). One of the important highlights of this model is the separation of the structural and functional components of the I&C components as Physical Breakdown Structure (PBS) and Functional Breakdown Structure (FBS) respectively. In PBS, highest level of organization is termed as Plant System (PS), which holds physical components. Physical components are generically defined and are simply tagged as one of the many standard types defined by the ITER project. Only the part of the physical components in the I&C system that deal with sensor or actuator signals are captured as part of this definition. These physical signals are specified as part of respective components and are interfaced with rest of the system through slow and fast controllers. These controllers are part of both PBS and FBS. Functionally, the ITER system is decomposed into three levels 'FBS Level 1' - maps to coarser plants systems at highest level, FBS Level 2 - maps to coarser I&C coordination, and FBS Level 3 - maps to individual I&C function such as temperature control and so on. Each I&C function has one or more functional variables which are entities with lowest granularity. A variable captures additional description such as alarm and archiving details, and needs to be either mapped to a physical signal or a

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-139-7

computation. Variables too are deployed on control units such as fast controller or plant system host (PSH). For any given FBS level 2, a plant system I&C (PSIC) entity is defined and is represented by one more control units, but only one PSH. PSH is also responsible for providing a generic or abstract interface of a plant system to the rest of ITER which it does through deriving global states of the plant system, e.g. if the system is up or down or in maintenance. To cater to the need of a PSIC to coordinate across plant systems, a higher level entity 'I&C Project' has been introduced to act as a place holder. Realization of this model consists of an entity-relationship model, its implementation as a relational database schema, a collection of Java classes with mapping to tables of this schema and domain rules to check consistency of the data. Operations on data are controlled by API exposed by the model implemented in Java. The API's implement abstract interfaces so that the underlying implementation can change in due course of project execution and allow smoother evolution of model and its consumers. There are project and domain specific rules built into the model that validates the data provided by the user and notify in case of issues that require corrections. As per convention, only a correct SDD specification can be used for code generation. The model also supports capturing implementation platform-specific information as well.

User Interface: SDD editor is available both as a standalone Eclipse based product and also as a web-based based application. Using the SDD editor, a user can populate self-description data of plant systems, which gets pushed into the model and then to the database through model APIs. Consistency checks are built inside the editors which can be invoked independently by users. The editors facilitate in code generation, navigation as well as retrofitting the changes back into model.

Code Generation: Code is generated as part of 'Forward Engineering' process. The code generation module refers to populated SDD, checks for consistency and finally generates source code and configuration files for various platforms that actually realize the ITER M&C system 'CODAC'. The platforms include 'EPICS', 'Control System Studio' (CSS), 'Siemens Step7 PLC development environment' and 'RedHat Enterprise Linux' (for shell scripts). Each platform require generation of multiple artifacts - usually text files - called as 'targets', each target having its own 'Platform-specific Model' (PSM) and associated transformations - from PIM to PSM model-to-model (M2M) transformation - and - from PSM to text model-to-text (M2T) transformation. To ease the latter transformation, text based templates are defined with appropriate blanks to be filled by PSM instance. A template engine evaluates these templates for a given PSM instance and for ITER SDD. Apache Velocity has been chosen as the template engine/framework for this.

Retrofitting: Artifacts generated as part of forward engineering can be enhanced or modified using their respective platform provided tools. However, these changes are lost when code is re-generated. To tackle this problem when it is semantically possible for some of the important platforms such as EPICS, the technique of retrofitting has been employed. It involves parsing of text files using available or specifically written parser routines to identify changes, which are then pushed back into the model. Changes such as deletions or renaming are not retrofitted and they need to be performed from either of the editors.

Handling Complexity

Some of the challenges with respect to complexity that the SDD approach deals with are mentioned below:

Scale: Mechanism to verify and handle specification of large volume of Self-Description data(>=1000 variables) [6][7]. Feature to import bulk data from external sources solves this problem.
Geographically Distributed Development: Mechanism to integrate components developed across geography. Implements a central SDD repository where all the created specification is stored [7][8].
Diversity: Mechanism to selectively retarget the SDD specification into target technology specific platforms. Support through the implementation of translator plugins and code generation templates [6][7][8].
Connectedness: As complexity of SDD specification grows due increased interdependency, it becomes challenging to verify the correctness of the target behavior. SDD tools facilitate this through incremental addition of domain specific rules [6][7][8].
Evolution: The complexity due to the long lifetime of the project creates stringent design requirements which mandate modularity, reusability, flexibility and adaptability to newer target technologies [7].

FUTURE ENHANCEMENT

The experience at ITER helped us conceptualize an integrated specification driven environment for building M&C solution that could be used across all similar projects such as SKA and help face similar challenges due to scale, complexity, diversity, safety criticality, reliability, availability, response times, data volumes, and timelines and so on. The section below provides our thoughts on how this approach can be enhanced.

Complete M&C Domain Model

Taking the ITER specification model as the basis we tried to identify and generalize the various types of specification elements that the M&C domain model will need to hold:

- **Control logic:** List of commands and parameters, responses from command execution, validation rules, FSM control logic, control actions & scripts, command distribution logic, response aggregation logic, state management specifications.

ISBN 978-3-95450-139-7

Copyright © 2014 CC-BY-3.0 and by the respective authors

- **Communication:** Addresses and ports, communication protocols and so on.
- **Data acquisition and processing:** Data acquisition specifications, validation rules, worldview database schema, subscriptions, feedback control, logging & archiving specifications.
- **Events handling:** Event and alarm detection rules, events and alarms acquisition specs, filtering of events and alarms, alarm handling and propagation specifications.
- **User interface specifications:** Data elements, display widgets, layout, behaviors.
- **Safety and security specifications**: Threat detection and response rules, authentication and authorization, data protection and network security configurations.
- **Reliability and availability:** Fault detection and handling, configuration of mechanisms such as heartbeats, watchdogs and failover.

The intention is to capture the consistency relationships and dependencies across these specification items failing which may result into various inconsistencies.

The implementation of this domain model is a meta modeling task which is done by synthesizing the specification modules based on different existing methodologies such as SysML and standards such as RBAC[9] for security and so on. The model also is augmented with capturing consistency relationships.

Specification Environment

Our belief that the specification environment would need to support the typical M&C development life cycle which are as described below:

- Define the controller requirements. Identify the context, the physical system and interfaces between themas a set of parameters and behaviors.
- Decompose the M&C system into subsystems, typically following the hierarchy of decomposition of the physical instrument. Allocate the controller requirement among subsystems, determining the functional and engineering responsibilities of each subsystem (e.g., reliability, performance, safety etc). Capture these as blackbox specifications.
- Design how the controller coordinates subsystems and provides services such as coordination and alarms handling to realize the overall system requirements.
- Iterate the above steps for each subsystem until no further decomposition needed.
- Provide tips, hints, warnings, errors and best practices within and across functional and non-functional units.

The implementation of such an environment is based on the following principles:

- Usage of standards for each aspect of the specification model, wherever applicable instead of inventing representations.
- Modularization of specifications, so that one module could be replaced with a newer standard without impacting the overall framework.
- Identification of mutual consistency constraints among the specifications modules for different aspects, and checking them during specifications creation. These consistency relationships need to be updated whenever a module is updated, or even if a new technology platform is chosen that creates additional consistency relationships.
- Capability to support retargeting to newer technology in the specification model.
- Exploitation of existing tools and technologies to minimizing effort for implementation, user learning and so on.
- Usage of the right metaphorsto capture structural and behavioral details visually or textually.

ACKNOWLEDGMENT

Our thanks to the ITER CODAC team for their support while building the SDD Editor. Thanks to Harrick Vin for his support that helped us pursue this problem further.

REFERENCES

[1] J.B. Lister, J.W. Farthing, M. Greenwald, I. Yonekawa: The ITER CODAC conceptual design, Fusion Engineering and Design 82 (2007) 1167–1173.
[2] A.R. Taylor, "The Square Kilometre Array", Proceedings IAU Symposium No. 291, 2012.
[3] Cleland-Huang, Jane (2012). Software and Systems Traceability. Springer. p. 48. ISBN 978-1-4471-2238-8.
[4] Object Management Group, “SysML Specification V1.3,” http://www.omg.org/spec/SysML/1.3/
[5] About EPICS, http://www.aps.anl.gov/epics/
[6] W. Mitchell Waldrop: Complexity - The Emerging Science at the Edge of Chaos (1993).
[7] Grady Booch: Object-oriented Analysis and Design 3E (2007).
[8] Erik W. Aslaksen: Designing Complex Systems - Foundations of Design in the Functional Domain (2012).
[9] D. Ferraiolo, R. Sandhu et al, RBAC Standard paper: TISSEC), 4(3): 224-274, Aug. 2001.

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-139-7