Migration of Supervisory Machine Control Architectures[*]

Bas Graaf
TU Delft
The Netherlands
b.s.graaf@ewi.tudelft.nl

Sven Weber
ASML and TU/e
The Netherlands
sven.weber@asml.com

Arie van Deursen
CWI and TU Delft
The Netherlands
arie.van.deursen@cwi.nl

Abstract

We discuss a first step towards an approach for migration of supervisory machine control (SMC) architectures. This approach is based on the identification of SMC concerns and the definition of corresponding transformation rules.

1. Introduction

The effort involved in modifying software systems can be reduced by, e.g., separation of concerns, the introduction of product-line architectures and model-based development. Adopting such techniques typically requires architectural changes.

In this paper we consider the migration of a supervisory machine control (SMC) architecture from a finite state machine (FSM) to a task resource (TRS) based paradigm. The latter can be implemented using a product-line approach.

We discuss the definition of architectural mappings, which relate realisations of concerns in the old to the new architecture. This is a first step towards an approach that reuses design decisions with respect to system behaviour incorporated in legacy (FSM)-based designs for the migration of an SMC system to a TRS-based product-line architecture.

2. Supervisory control

For the execution of manufacturing requests by an advanced manufacturing machine, such as an ASML wafer scanner, multiple alternatives exist with respect to manufacturing activities and associated mechatronic subsystems. This raises concerns such as the selection and controlled use of mechatronic subsystems, state consistency of these subsystems for consecutive activities (e.g., setups), and the conditional and concurrent execution of activities. An SMC system addresses these concerns.

FSM-based supervisory control This approach is proposed by, e.g., Ramadge and Wonham [1]. Here, the set of possible machine behaviours is considered to form a language. A discrete supervisory FSM is synthesised that restricts this language by disabling a subset of events to enforce valid machine behaviour. Here, all decisions with respect to the aforementioned concerns are taken design-time. As such, multiple FSM's are used per controller that are synchronised through (shared) events or variables. This significantly reduces modifiability and reusability. In industrial settings UML state machine diagrams are often used to specify these FSM's.

TRS-based supervisory control In this approach, tasks correspond to manufacturing activities and resources correspond to mechatronic subsystems. The TRS-based approach as proposed by Van den Nieuwelaar [2] allows for the introduction of a product-line architecture. Figure 1 reflects a module-view of this architecture.

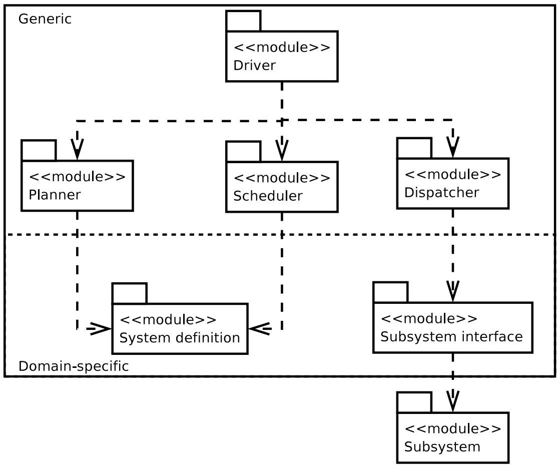

Figure 1. The product-line SMC architecture

In this architecture, generic and reusable components are introduced for rule-based planning (determine alternatives), scheduling (select task-resource combinations) and dispatching (release tasks to resources for run-time execution). The involved rules interpret manufacturing requests and operate on resource types (capabilities) and task types (behaviours). This architecture offers variability with respect to tasks and resources, and can be instantiated for a domain by implementing a few domain-specific modules.

[*] Partial support received from SenterNovem, project Ideals and Moose, and from NWO Jacquard, project Reconstructor

3. Approach

We consider the migration from an FSM to a TRS based architecture as a transformation between source and target architecture views. In our case the source view are depicted as UML state machine diagrams and the target views contain a specification of the domain specific part of the product-line architecture of Figure 1. In order to define a reproducible mapping between source and target views, we define transformation rules on the level of the associated viewpoints (meta-models). For practical reasons, a normalisation step is introduced that results in a canonical source view. The meta-model of this normalised view is obtained by augmenting the source meta-model with SMC specific constructs and enforcing well-formedness rules. Next, transformation rules are applied to the canonical source view, resulting in the target view.

For the target view, depicting the domain-specific part of the product-line architecture (Fig 1, System definition), we introduce the meta-model as shown in Figure 2. This meta-model is defined using the Eclipse Modeling Framework (EMF) [3] and is based on the approach proposed in [2] and some practical SMC issues.

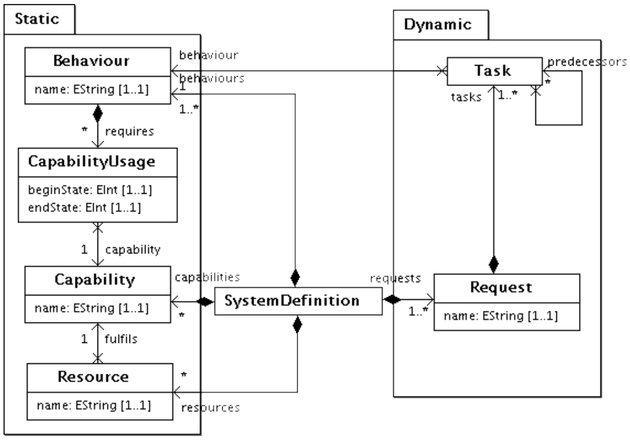

Figure 2. Target meta-model

The static part of the SystemDefinition defines the available Behaviours, Capabilities and Resources. In addition, it defines which Capabilities are used for which Behaviour and the corresponding beginState and endState of the involved Resource (CapabilityUsage). When considering the source view, an action with associated completion event typically translates to a single behaviour. Similarly, a list of unconditional, event-independent actions can be aggregated to form a single behaviour. The identification of capabilities and resources is less intuitive. Coarsely put, the presence of a mutual exclusion mechanism, combined with a function call that triggers a state transition, indicates resource usage. In turn, capabilities are identified based on resource similarity. The context in which these mechanisms are encountered define the begin and end states. This information is used to, e.g., determine the necessity of setups.

The dynamic part of the SystemDefinition defines how to map a manufacturing Request to Tasks (of a specific Behaviour) and assign Resources (of a required Capability). Constraints on Task synchronisation are defined using the predecessors association between Tasks, which can be derived from the source view by taking into account the execution order of actions. Concurrency, indicated in the source view using concurrent states, is also constrained by the predecessors association.

As such, the meta-model currently only takes into account planning rules, leaving out scheduling and dispatching rules. These will be added once we have implemented the transformation rules we have defined until now.

4. Conclusion

In this paper we proposed a generic, two-phased transformation approach that can be applied to the migration of supervisory control systems without loss of generality.

Having defined the meta-models using EMF, we made a first step towards tool support for architecture migration, both in terms of code generation and automatic transformation. Currently, we are implementing the transformation rules, which were defined informally so far, using the Atlas Transformation Language (ATL) [4]. The ATL transformation engine can be used with EMF (e.g., Figure 2) or MOF (e.g., the UML meta-model) meta-models.

A next step will be to extend the approach to include the legacy source code as a starting point for automated reconstruction and migration. In turn, this would require a more formal definition (e.g., using ATL) of the transformation rules. Although, this approach would be more domain specific than the one described in this paper, such an automated approach to migrate SMC systems from an FSM-based architecture to TRS-based product-line members is highly desirable in industry.

References

[1] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

[2] N.J.M. van den Nieuwelaar. *Supervisory Machine Control by Predictive-Reactive Scheduling*. PhD thesis, Technische Universiteit Eindhoven, 2004.

[3] Eclipse Foundation. Eclipse modeling framework (EMF). `http://www.eclipse.org/emf`, 2005.

[4] ATLAS Group. The ATL home page. `http://www.sciences.univ-nantes.fr/lina/atl`, 2005.