Behavior Engineering for the Analysis and Verification of Processes

David Tuffley$^{1,2, a}$, Lian Wen$^{1,2, b}$

$^1$School of Information & Communication Technology, Griffith University, Brisbane, Q4111, Australia
$^2$Institute of Integrated & Intelligent Systems (IIIS), Griffith University, Brisbane, Qld. 4111, Australia

$^a$d.tuffley@griffith.edu.au, $^b$l.wen@griffith.edu.au

Keywords: Behavior Engineering, process modeling, process verification

Abstract. Engineering in general is a process-intensive domain. Processes and process models underpin the activities of every Engineering discipline, ensuring consistent outputs by sequencing the right activities in the correct order. Acknowledging the importance of process models, there is a clear need for effective methods for verifying such models to ensure efficiency and correctness. This paper introduces a new formal methodology for the analysis and verification of processes, Behavior Engineering (BE) [1]. BE has proven very effective when applied to the development of process models. In this example, the discipline is Software Engineering, yet it is argued that the approach can be applied to any process or set of processes from Chemical, Materials and Metallurgical Engineering.

Introduction

This paper outlines how Behavior Engineering (BE) method [1] can be applied to disambiguate processes and process models. Process models in this context are defined as a structured collection of processes that combine to achieve a desired outcome. BE has the capability to clarify terms, identify duplications and errors, and generally make a process simpler and more efficient. Also, as a formal method, BE models could be simulated [12] and model checked [13].

It is argued that the Behavior Engineering method has broad applicability across Engineering domains as a process verification tool.

More broadly, it is suggested that this method may be helpful when modeling processes at the project and organizational levels, including project-level processes covering a wide range of activities, including but not limited to project conduct, high-level policy documents, requirements specifications, and project agreements.

1. Process models

In general terms, process models are abstract representations of a process architecture, design or definition [2]. They are abstractions, not direct representations of reality and must not be mistaken for such. As George Box noted, all models are wrong but some are useful [3]. Models are simplifications of reality and in the process of simplifying, essential information might be left out resulting in ambiguity. Even with ambiguity, models can be useful, but this ambiguity points to a need for a way to reduce or eliminate it. Such a way might be found in a formal method such as Behavior Engineering.

In seeking a solution to the problem of ambiguity in process modeling, one sees a similar problem with the requirements specifications for software intensive systems which are themselves an abstract representation of a future computer system. Ambiguous language, incomplete descriptions, repetition and redundancies in the way specifications are expressed inevitably leads to sub-optimal outcomes.

2. Behavior Engineering

Behavior Engineering uses a formally-grounded graphical notation with the capability to represent a wide range of processes in unambiguous terms. Its strength is its ability to accommodate complexity
and detail, ease of use, and in particular for its ability to expose defects. Previous research indicates that BE can be useful verification tools for process modeling and verification [4].

In a broad sense, Behavior Engineering is a method for assembling individual pieces of requirements to form an integrated component architecture. Each requirement is translated into its corresponding behavior tree (BT) [12] and composition tree (CT) [14], an example of which is shown below, which describes unambiguously the precise behaviors of this particular requirement [5] or process. The ‘tree’ is built up from (a) components, (b) states the components become, (c) events and decisions/constraints associated with the components, and (d) the causal, logical and temporal dependencies associated with the component [5].

Duplications and redundancies are identified and removed, for example, when the same requirement is expressed twice using different language in different places. Another benefit is that traceability becomes more manageable.

The OOSPICE Project [6] had the overall aim of improving time-to-market, productivity, quality and re-use in software development by optimizing the processes and technologies of component-based software development (CBD). OOSPICE combined four major concepts of software engineering: (a) CBD, (b) object-oriented development, (c) process assessment and (d) software process improvement. Its objectives were the definition of (a) a unified CBD process metamodel, (b) a CBD assessment methodology, (c) resulting in component-provider capability profiles, and (d) a new CBD methodology and extensions to the ISO/IEC 15504 [7].

A key part of the OOSPICE was the definition of a model of coherent processes addressing the issues of component-based development; the process model was strongly aligned to ISO/IEC 12207 [8], but it incorporated significant additional processes that specifically addressed CBD issues. The process model was developed following the approach of ISO/IEC 12207 [8], with a series of structured activities and defined tasks. Additional detail on input and output work products was also specified for each activity.

The OOSPICE process model was examined using the behavior tree method in order to assess its consistency and completeness. The results should be viewed in the context of the OOSPICE process model having already been intensively peer-reviewed by the international working group that developed it.

Results. The results of using BE exceeded expectations. BE revealed no less than 73 task level problems, 8 process level problems and numerous task integration problems. In addition, instances were found where fragments of tasks were identified which subsequently have no integration point with the larger process tree – a weakness caused by unspecified or inconsistent task inputs or outputs.

Figure 1 shows the integrated tree resulting from analysis of a single process. It clearly shows missing (dark shaded 5.2.3.1, 5.2.3.4) and "implied" but unstated elements (light shaded, 5.2.3.1, 5.2.3.3, 5.2.3.5), and also a failure in integration, resulting in lack of overall consistency [9]. The medium-shaded boxes were unchanged.

Encouraged by success on OOSPICE, the technique was subsequently applied to the review of the Capability Maturity Model Integration (V 1.2) [10] in Figure 2. This work was undertaken in the context of a review of drafts of the Software Engineering Institute’s Capability Maturity Model Integration (CMMI) V1.2. The results of this analysis formed the basis of requests for change submitted to the SEI. In view of the resource restraints, it was not possible to apply the technique to support the complete review, but where problems were seen to exist, an analysis was conducted.

Figure 2 therefore demonstrates how the Behavior Engineering approach helped to clarify an ambiguity in the specification of the Requirements Development Process Area in the Capability Maturity Model (Integration). Again, the two dark-shaded boxes indicate missing material, while the extreme left and bottom right boxes "implied" but unstated elements.

It can be seen from the Figures 1 and 2 that the Behavior Engineering approach can be applied to the verification of process models, even large, complex models.
7. Conclusion

The integrated, graphical approach of Behavior Engineering has proven itself in Systems and Software Engineering when developing process models [4] [9] and large-scale software-intensive systems [1]. Arguably, the generic nature of BE makes it a valuable tool for Chemical, Materials and Metallurgical Engineering or indeed any other that relies on the rigorous application of process.
Behavior Engineering’s strength is in the innovative way that it addresses the problems of scale, complexity, and imperfect knowledge associated with the complex systems [11].

Employing a graphical Behavior Modelling Language (BML), a coherent behavioral, compositional and structural integrated view of the process model or system is developed based on natural language descriptions of a process or system. These three formalized views extract all of the useful information that is needed to preserve and clarify the intention expressed in natural language. A Behavior Modelling Process (BMP) is used to construct the integrated views [11].

It is suggested, based on the discussion above, the Behavior Engineering is a useful tool for process developers in the broadest range of Engineering disciplines, particularly where there is a need to bring coherence and order to a large set of unstructured or semi-structured information.

References


