Requirements-Reuse Using GOPCSD: Component-Based Development of Process Control Systems

Islam A.M. El-Maddah and Tom S.E. Maibaum

King’s College London, Computer Science Dept., London WC2R 2LS, UK
Islam-elmaddah@ieee.org, tom@maibaum.org

Abstract. Software reuse, when correctly employed, can make it feasible to extend process control applications with controlled cost and effort. Component-based development is one of the important means to realise software reuse at the different development lifecycle stages. This paper illustrates the component-based development of process control systems using the GOPCSD tool. The GOPCSD tool guides the user to develop flexible requirements specification models for process control components that can be reused in different families of process control applications. The tool automatically generates a B specification corresponding to the corrected requirements. We illustrate the component-based development by examining a case study of a production cell. Finally, we draw conclusions and give directions for future work.

1 Introduction

Component-based development has gained considerable attention as one of the major software reuse directions. This development process involves identifying, building and testing software segments to be reused in similar applications from one domain [25]. Since, software development activities cover a broad area ranging from problem understanding, requirements, specification, design, implementation and testing [20 p6], the component-related issues may be targeted at the different development lifecycle stages. Still, most of the effort has been devoted to developing high-level languages libraries. These libraries usually contain implementation details and are tailored for a specific programming language, such as ADA or JAVA. As a result, reusability will be restricted to a single implementation language, as well as a potential effort is required to trace such implementation code segments back to the requirements and specification levels.

A significant research has focused on the software reuse at the pre-implementation levels, as follows: Jackson, in [16], suggests reusing problem patterns to understand and decompose complex user needs. Whereas in [27], Massonet and Lamsweerde use query generalisation to check the analogy between similar systems and then use formal rules to elaborate the requirements for the derived systems. Reubenstein and Waters, in [30], use specialisation of existing systems requirements to create new systems. At specification level as in [24], Maiden and Sutcliffe utilise analogy between similar
systems to import specification segments. Whereas in RSML [22], the idea of building software specification models of process control components has been evolved to a level of building state charts for the components and model their interaction using higher-level state charts. Other efforts have focused on software components at various development lifecycle stages, such as [2, 3, 4, 6, 13, 14].

Process control systems range from in-house appliances to industrial sophisticated plants (such as Oil Refiners). In process control, systems are built out of physical components, such as robots and tanks. Specifying these components constitute the starting point of developing new control applications or extending existing ones. Then, by interconnecting the components properly, the overall control system will function efficiently and meet the client needs. Understanding the importance of the requirements analysis [28], a requirements model for the process control components should be developed at such early levels, as in RSML [22], where state charts model are used as a means of formal requirements specification.

Formal methods have been used to develop reactive systems. These methods enable the user to develop an implementation that conforms to the early specification. In particular, the B formal method [1] has refinement, hierarchical and abstract state foundations, which make it suitable to specifying systems that are built of sub-systems such as process control systems. Moreover, the existing tools, such as B toolkit [5], which supports B, create a rigid route that traces the B formal specification to the implementation stage, such as JAVA. However, the level of understandability and usability of the B method, is not adequate for a client, such as a process control systems engineer. Because, this entails dealing with the sophisticated mathematical and logical nature of the formal specifications; this creates interference of concerns between requirements and specification [23].

Thus, we were motivated to start component-based development as early as at the requirements level by adopting the goal driven requirements analysis method of KAOS [17], as a starting point. The hierarchical structure of a goal model provides a feasible means for tracing user needs and refining them gradually, as noted in [8]. A requirements analysis tool, GOPCSD (Goal Oriented Process Control Systems Design) [10] has been developed for the specific domain of process control systems. GOPCSD provides a dynamic requirements library, which supports reusable components and general goal templates. Furthermore, the tool automatically translates the complete and satisfactory requirements to formal specifications in B, which can be in turn translated to high-level languages, within the B toolkit environment or similar environments; hence, the stage is prepared for the software engineer to manipulate the application from the software design and architecture point of view.

In this section, we have briefly focused on the research area. In section two, we describe our implementation and adaptation of goal-oriented requirements analysis of process control systems. In section three, we illustrate the component-based development of the GOPCSD tool. In section four, we examine a recurring example in the literature of a production cell [21] to illustrate integrating application from prefabricated and trusted components. In section five, we describe briefly the process of generating B machines. Finally, in section six, we draw the main conclusions and give suggestions for future research directions.