The GOPCSD Tool: An Integrated Development Environment for Process Control Requirements and Design

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

Department of Computer Science, King's College London
London WC2R 2LS, UK
{elmaddah,tom}@dcs.kcl.ac.uk
Fax +4402078482851

Abstract: The GOPCSD (Goal Oriented Process Control Systems Design) tool is an integrated environment, where the process control systems engineer can construct, import, check, reason about, modify, validate requirements specifications and generate in the B specification language a formal specification of such process control requirements. Borrowing from the KAOS method, the GOPCSD tool adopts the goal-oriented hierarchy concept to enable easy tracing of the user needs to the requirements level, as well as the requirements to the design specification level. The tool offers a library and formal and informal checks and tests to aid correction and enhancement of the requirements; in addition, the normal systems engineer can use the tool effectively and automatically generate a B formal specification, thus not demanding a high-level of knowledge about the sophisticated mathematics supporting formal methods like B.

1 Introduction

In reactive systems, such as process control systems, where safety and security are considered important aspects, along with the system's operational constraints, the B formal method has been demonstrated to provide effective support at the early design stages. However, a requirements stage before the formal method is still needed, concerned with the construction, reuse, correction, modification, testing and enhancement of the requirements.

In addition, tracing the user needs to the requirements level, as well as tracing each requirement to the design specification level decreases the gap between the user's perspective and the specification level. This motivated us to adopt the goal-oriented requirements analysis method of KAOS [3] as a starting point. The goal driven requirements analysis method of KAOS uses goal-models as the formal model to structure the software requirements gradually and precisely; the main goals of the goal-model usually represent the user needs, while the low-level goals will be translated to specification building blocks. This ensures the traceability goal as well as the understandability of the goal-model by the user.

Thus, we were motivated to start the requirements analysis as close to the view of the systems engineer as in [8] and, furthermore, to extend the formalisation, via an automated tool, to the B specification level [1], as in [6, 9]. In this paper, we introduce the GOPCSD tool that serves as a front end for the B toolkit [7] or other environments that adopt B, to hide the mathematical details of the B method and to allow the systems engineer to focus on providing precise and formal requirements, while preparing the stage for the software engineer to use the generated B specification to produce code that is not only correct with respect to the high-level B specifications, but is also much more likely to satisfy the systems engineer's stated requirements.

2 The GOPCSD Tool

The GOPCSD tool [2] (as shown in fig.1) is designed to analyze the requirements of process control systems and automatically generate B formal specifications. To build an application within the tool, there are three main phases. In the first phase, the requirements can be constructed using different (process control specific) entities (components, agents, variables, goals, goal-models, and variable types). The second phase checks the consistency, completeness, reachability, and the validity of the goal-model. The tool suggests different goal-model modifications to resolve goal-conflict or unreachable goals, avoid obstacles, or complete the requirements. Thus, the first and second phases can be regarded as a feedback loop to devise consistent and complete requirements. Finally, after the goal-model is considered satisfactory, the application proceeds to the third and final phase producing automatically the software specification through translating the goal-model to a B specification. The GOPCSD tool hides the details of the B language from the systems engineer, and increases the separation of concerns between the software and systems engineers.

Although the GOPCSD method is based on KAOS, there are some significant adaptations of the KAOS method to fit with the nature of Process Control Systems and to enable an automatic phase to generate B specifications to be added. The tool restricts the variables of the application to have finite domain in order to enable feedback for the completeness, consistency and validity of the requirements.

2.1 GOPCSD Tool Support for Reusability

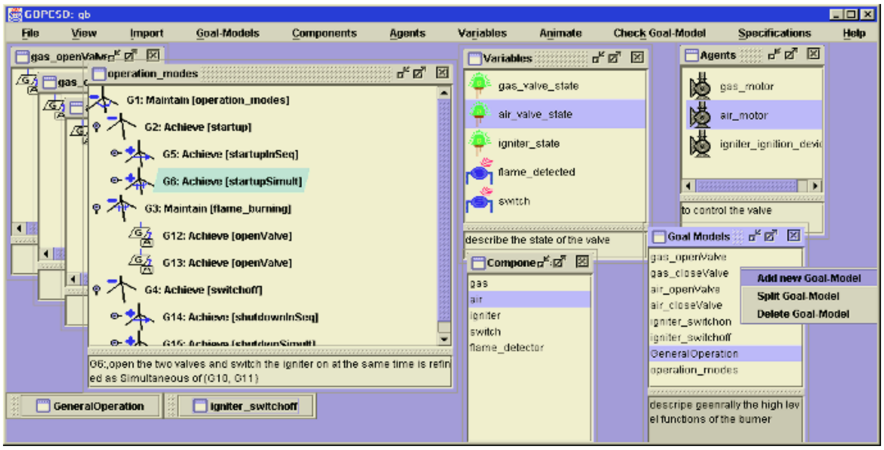
Since the main target of the GOPCSD tool is process control systems, this motivates a need for building systems from sub-systems and components. Therefore, the GOPCSD tool supports reusability in building the goal-models based on two basic concepts: similar applications are built of the same kinds of physical components, and similar applications can have similar high-level goals, even if they have different components.

The first concept suggests a library for the frequently used components with their associated low-level goal-models to enable the user to import them into his/her applications. The second concept recommends high-level goal-model templates that can be used in similar systems with possibly different component details. For example, a production cell with a single press can have some of the same high-level goals as a double-press production cell.

2.2 Constructing Goal-Models (Phase I)

The GOPCSD method mainly depends on reusing elements of the library to increase the reusability and the maintainability of the developed applications. Thus, the design starts by importing components from the library in addition to the high-level goal-model templates. After the systems engineer decides on the components and/or the high-level goal-model templates to be imported, a re-naming and/or mapping process can take place in order to change the components' or the templates' variables and/or agents' names into the corresponding ones to be used in the new application.

Then, the systems engineer formulates the application's main goals; these generally can be considered as medium-level goals between the high-level goal-model templates and the low-level component goal-models. The main goals can be placed in separate empty goal-models until the user links them by combining them into a higher-level goal. The process of producing a single complete goal-model requires the user to link the high-level goal models, the main goals and the low-level goal-models of the components.



G1: Maintain [operation_modes]
G2: Achieve [startup]
G5: Achieve [startupInSeq]
G6: Achieve [startupSimult]
G3: Maintain [flame_burning]
G12: Achieve [openValve]
G13: Achieve [openValve]
G4: Achieve [switchoff]
G14: Achieve [shutdownInSeq]
G15: Achieve [shutdownSimult]

G6:,open the two valves and switch the igniter on at the same time is refined as Simultaneous of (G10, G11)

gas_valve_state
air_valve_state
igniter_state
flame_detected
switch

describe the state of the valve

gas_motor
air_motor
igniler_ignilion_devic

to control the valve

gas
air
igniter
switch
flame_detector

gas_openValve
gas_closeValve
air_openValve
air_closeValve
igniter_switchon
igniter_switchoff
GeneralOperation
operation_modes

descripe geenrally the high level functions of the burner

Fig. 1. The desktop of the GOPCSD tool, showing the requirements elements (components, variables, agents and goal-models lists) to the right and a goal-model to the left.

2.3 Checking the Goal-Model (Phase II)

The GOPCSD tool enables the user to perform various checks and tests to enhance the requirements and to detect any incorrect or undesirable behaviours implied by the requirements, as early as possible. The tool offers the following checks:

Checking Correctness of Goal-Models. There are some essential constraints the tool enforces on the goal-models in order to build correct B specification (and correctness is defined with respect to these constraints and those pertaining to the structure of goal trees). These constraints address the basic definitions of terminal goals, variable controllability, and refinement patterns. This utility will highlight the goals that violate the constraints.

Reasoning and Investigation Utilities. The GOPCSD tool enables the user to reason about the how and why of the different goals of the goal model. Reasoning how to achieve one goal lists the sub-goals of this goal and their sub-goals; while reasoning why to achieve one goal ascends the goal-model level by level listing the details of the ancestor goals. The reasoning utility can be regarded as an informal early level of checking to validate parts of the goal-model, on the one hand; on the other, it can be used to guide the user to elicit new goals to complete the construction of the goal-model, either upward answering why or downward answering how [3]. Another utility offered by the tool is highlighting the goals containing a specific variable or the goals that are affected by a specific agent. This utility is similar to dependence graphs, or tracing utilities, which can be helpful to judge variable and agent coupling.

Checking the Reachability of the Requirements. Another important check that is helpful to amend the requirements is detecting unreachable goals. The user usually errs in specifying the conditions of some of the goals or locates them in inappropriate positions where they will never be activated. Thus, the goal-model can be valid, complete and consistent but some of its goals' pre-conditions can be unreachable.

Checking Completeness. Completeness means that for each combination of the application's variables there is a defined action(s) to be taken. Some incompleteness can occur as a result of ignoring unexpected variable combinations or ignoring variable combinations under specific situations. The systems engineer may choose not to include such combinations for normal operation, but during the application execution they might occur and produce a hazard or incorrect operation.

Checking Obstacles. An obstacle is a sequence of events that can occur during application run-time and can obstruct some of the goals from being achieved [4]. Obstacle analysis can be performed at the lowest level within the complete goal-model, when all the terminal goals have a complete formal description. Each formal description of a terminal goal will be negated in turn and the user attempts to find cases that validate the negated conditions and, hence, the goal-model can be modified in order to prevent the obstacles from occurring or to attenuate their effects.

Checking Goal Conflict. Conflict arises when two or more goals prescribe the performance of inconsistent actions under the same conditions [5]. Conflicts are checked by comparing the conditions for goals that assign different values to the same variables. When a goal-conflict is detected, the tool suggests that one of the conflicting goal's formal pre-conditions should be modified. This utility can be employed to bring together the various process control requirements aspects, such as safety, security and productivity. In case of conflicts, the GOPCSD tool will suggest to modify the operational goals in order to pay attention to the various issues.

Animating Goal-Models. The systems engineer usually needs to validate the requirements. Such validation should emulate the execution of the controller during run-time. The GOPCSD provides different utilities, such as saving, loading and executing event lists, emulating output delay and faults, and displaying the description of the activated goals, cycle by cycle. This validation utility provides the user with an easy to use means of understanding the requirements model. In addition, it guides the user to fine-tune the goal-model of the application to enhance the requirements and remove bugs as early as possible.

3 Translating Goal-Models to B Machines (Phase III)

After the user validates the goal-model and approves the description of the controller, the tool translates the corrected requirements into formal specifications represented as B machines. The translation from goal-models to B specifications consists of two steps, as follows:

3.1 Splitting Compound Goal-Models

Alternative refinement is used to enable the user to express more than one alternative solution for the control application. To reduce the effort expected form/by the user, the shared parts of the compound goal-model are available for each version. However, to build controller specifications, only a single solution can be used. Thus, the translation into B specifications starts by splitting the goal model if it has alternative refinement sites.

3.2 Translating Goal-Models to the B AMN Language

The complete goal-model, which represents a separate solution, will be automatically translated by the tool to a main controller B machine and a number of actuator B machines, depending on the number of the active agents used within the goal-model. In addition, the definitions of the various data types of the variables will be collected in a data type B machine. The different values that can be assigned to the output variables will be represented as operations of the actuator machines. The terminal goals of the goal-model will be grouped by the output variable they control and will be translated to parallel parts of one operation of the main controller machine; this will ensure that the controller can manipulate the output variables in parallel.

References

1. J. R. Abrial, The B Book: Assigning Programs to Meaning, Cambridge University Press, 1995.
2. I. A. El-Maddah and T. S. Maibaum, Goal-oriented Requirements Analysis of Process Control Systems, first MEMOCODE, France, 2003

3. A. Van Lamsweerde, A. Dardenne, B. Delcourt, and F. Dubisy, The KAOS Project: Knowledge acquisition in automated specifications of software, proceeding AAAI Spring Symposium series, Track: Design of Composite Systems, Stanford University, March 1991, pp 59-62.
4. A. Van Lamsweerde and E. Letier, Obstacles in Goal-driven Requirement Engineering, proceeding ICSE'98 20th international conference on software engineering, Kyoto, ACM-IEEE, April 1998.
5. A. Van Lamsweerde, R. Darimont and E. Letier, Managing Conflicts in Goal-Driven Requirement Engineering, IEEE Transactions on Software Engineering, Special Issue on Managing Inconsistency in Software Development, Nov. 1998.
6. K. Lano, K. Androutsopoulos, and D. Clark, Structuring and Design of Reactive Systems using RSDS and B, FASE, ETAPS 2000
7. K. Lano and H. Haughton, Specifications in B, An introduction using the B toolkit, 1996, Imperial College Press
8. E. Letier and A. V. Lamsweerde, Deriving Operational Software Specifications from System Goals, SIGSOFT 2002/FSE-10, Nov. 18-22 Charleston, SC, USA.
9. E. Sekerinski, Graphical Design of Reactive Systems, D. Bert (Ed.) 2nd International B conference, Montpellier, France, Springer-Verlag, 1998.