Copyright © 2002 IFAC
15th Triennial World Congress, Barcelona, Spain

MODEL CHECKING IN PATTERN BASED CONTROL SYSTEMS DESIGN.

Jüri Vain and Juhan Ernits

Department of Control Systems, Institute of Cybernetics at Tallinn Technical University, Tallinn

Abstract: The idea of architectural and behavioural patterns originating from OO design community is applied to control systems design modeling and verification. A template for specifying modeling patterns is defined and, as an example, the Control Component Pattern (CCP) is proposed. The benefits from parameterization and abstraction encoded into the pattern are shown, allowing to increase the size of models and verification tasks that still remain efficiently decidable by model checking. The interchange between CCP and a certain subset of Simulink models allows to apply model checking in parallel to quantitative simulation techniques. A sketch of the application of CCP for a simple temperature control system design modeling and verification is presented. *Copyright ©2002 IFAC*

Keywords: modeling, pattern, components, verification, timed automata

1. INTRODUCTION

Design of modern control systems sets high requirements to system performance, safety, fault-tolerance and reliability. The major challenge in system development process is ensuring the correctness of the design at the earliest stage possible. Such effort is increasingly becoming more significant in the development cycle and, more generally, in the budget. Often the most resource-consuming part is not the search for correctness proof by some verification tool but the definition of the system design model and requirement specifications to be satisfied by the model. Proving or refuting satisfaction relation between the model M and requirements specification R is called model checking (MC) (formally stated as $M \models R?$ problem). Regardless of remarkable progress in MC techniques, e.g., systems with more than 10^{120} states have been reported verified in semiconductor and processor industy (Clarke *et al.*, 1999), several case studies (see (Hune *et al.*, 2000)) have shown that complexity issues are still major obstacles in using MC for industrial size systems. Efforts to be spent on MC are very much dependent on particular application, specification style, methodological framework of design and people's skills .

An approach, proposed in this paper for handling industrial size MC problems, is constructing design models and correctness formulas using domain specific modeling patterns and specification schemes.

General purpose program design patterns are well-known in OO design community already since mid 1990s (Gamma *et al.*, 1995)). In the design of control systems some extra aspects must be represented in modeling patterns and in requirement specifications: for verification of behavioral properties the hybrid dynamics has to be presented explicitly, and if safety critical applications are considered, fault tolerance properties have to be related to the functional specification. The Control Component Pattern introduced in this paper for modeling and verification of control system designs is an abstraction of typical control system components such as sensors, controllers, actuators and their compositions.

Another problem related to efficiency of MC is parametrization of patterns allowing application dependent model tuning and selection of appropriate level of model granularity. Having *a priori* complexity estimates for a given pattern one can find a proper trade-off between modeling details and decidability of MC problems without wasting time on too abstract models or hopelessly complex tasks.

The paper consists of following parts. In section 2, a general template for specifying modeling patterns is presented and the CCP is defined. In subsections 2.2 and 2.5, properties that can be explicitly modelled and verified by CCP are discussed. Section 4 explains implementation of CCP using XML technology that provides an easy way for model interchange between model checking and simulation tools. An example of application of CCP for temperature control system design is presented in 5. The benefits of and open questions about the proposed approach are summarized in the conclusion.

2. CONTROL COMPONENT PATTERN

2.1 Pattern template

Design pattern is a fundamental concept in OO design for almost ten years. Patterns name, abstract, and identify the key aspects of common design structures that make them useful for creating a resusable design (Gamma *et al.*, 1995).

Our goal is to adapt the idea of patterns for model construction and for model checking (MC) purposes to handle systematically the complexity of industrial size designs. Normally patterns are results of long evolution and accumulate the best practice in domain. Benefits expected from patterns in MC are following:

- Patterns provide abstract and structured description of modeling and related to that MC problems.
- They define cost, space and time trade-offs of applying alternative solutions
- A pattern can be attributed with characteristic MC tasks and their complexity estimates
- The model and MC strategies can be optimized with regard to standard MC tasks of given pattern.

The patterns have to be specified by uniform templates to simplify their reuse. We propose a template having the following structure:

- informal description (pragmatics)
  (1) *name* and *classification* (prerequisites for creating a pattern system or catalog)
  (2) *intent*: What particular problem does it address?
  (3) *motivation*: How does the pattern solve the problem?
  (4) *applicability*: What are the situations in which the pattern can be applied?
  (5) *consequences*: What are the trade-offs of using the pattern (what quantities are parameterized?)
- formal description
  (1) *structural definition* (UML class diagrams)
  (2) *behavioural definition* (timed automata/Petri Nets)
  (3) *MC query templates* (e.g., temporal logic TLTL formulae) and appropriate search strategies.

2.2 Conceptual definition and pragmatics of CCP

The CCP represents an external view on the control system components and can be considered as one possible (discrete time) approximation of the hybrid dynamical systems model proposed in (Koutsoukos *et al.*, 2000).

Generally, a component of CCP is defined as a triple

$$< \mathbf{I}, \mathbf{O}, R(\mathbf{I}, \mathbf{O}) >$$

where $\mathbf{I}, \mathbf{O}$ are respectively observable inputs and outputs (e.g., in practice, signals, disturbances, etc). $R(\mathbf{I}, \mathbf{O})$ is the set of (static and dynamic) input-output relations. An important simplifying assumption for modeling here is the independence of component's inputs. To achieve better modularity of the model, the input-output relation $R(\mathbf{I}, \mathbf{O})$ is split into a set of simpler relations R_j each corresponding exactly to one output O_j:

$$R(\mathbf{I}, \mathbf{O}) \equiv \bigcup_{j=1}^{m} R_j(\mathbf{I}, O_j)$$

where $\mathbf{I} = (I_1, ..., I_n)$ and $m = |\mathbf{O}|$. Defining $R_j(\mathbf{I}, \mathbf{O})$ for whole input domain is often very difficult if possible at all. Therefore, we introduce the notion of $mode$ denoting partitions on the input domain. Now, each k-th part of given partition (if properly selected) constitutes the domain for some local approximation R_{jk} of R_j such that R_{jk} is a total (for k-th part) and possibly linearizable function. Assuming that in each mode a different input-output relation holds and having p modes for an output O_j (input regions defining modes must not be convex), the behaviour of O_j is defined by an invariant

$$\bigwedge_{l=1}^{p} (g_l^j \Rightarrow R_{jl}(\mathbf{I}, O_j)).$$

where g_l^j denotes a l-th mode invariant for the output O_j and R_{jl} is the l-th transfer function of the component.

Regardless of its conceptual simplicity, CCP makes it possible to model a rather broad class of control system components including controller software and sensors with degrading effects (see for details (Vain and Kääramees, 2000)).

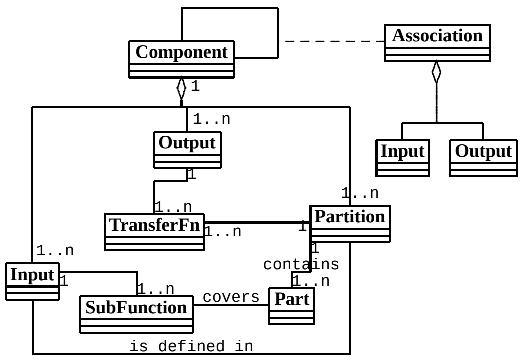

Figure 1. UML representation of the structure of the component.

To model real-time behaviour, the I/O relations and mode invariants are defined generally as timed expressions. Continuous components without switching have only one mode and a single transfer function for each output. Discrete components have a set of modes each with a constant transfer function. Symbolic input values, defined by mode invariants, constitute the input alphabet of the given component. Hybrid dynamics (in case of discrete time, continuous state) with switching (i.e., without jumps) expects that the transfer function has a member representing delay, i.e., the function value at some previous time instant. It can be easily implemented by means of feedback connection between the output and an auxiliary input, which keeps that output value until the next output value is provided. Hybrid dynamics means generally discontinuities in state trajectories. This kind of behaviour is approximated by the modes, some of which have constant and some of which have discrete time continuous state transfer functions.

The nonstationary behaviour can be easily modelled by defining transfer function parameters (e.g., coefficients or partition bounds) being auxiliary inputs. It means that they can be connected to outputs of some other component which models the dynamics of those parameters. On the other hand, it is obvious that redefining parameters as variables immediately brings in nonlinearity of I/O dependencies and as a consquence leads to undecidability of most of analysis problems.

2.3 Structural definition

The general structure of the pattern is depicted in the UML class diagram of Figure 1. Here pattern constituents and their relation types (association, aggregation, dependency) are shown. In fact, this diagram can be extended with refinement and aggregation relations between components when multilayered design structures are of modeling concern. Still, for defining operational semantics of CCP the flat structure suffices.

2.4 Behavioural definition

Integrated usage of simulation and model checking tools in control systems design presumes quite often transforming continuous models into their discrete counterparts. The reason for switching over to discrete approximations is efficiency and decidability of broader class of MC tasks compared to that of pure hybrid models. In the following we show key steps of defining CCP as a network of timed automata and discuss some of its properties.

Model structure. Each CCP component is realised by two automata templates represented in Figure 2 The first template "InputsOf" models the latency of inputs. Template "ModeBy" represents the mode selection by mode invariant conditions, and computing the value of the transfer function in selected mode. Thus, each CCP component is modeled by $p + 1$ structurally uniform timed automata, where p is the number of input partitions and equals to the number of ModeBy automata.

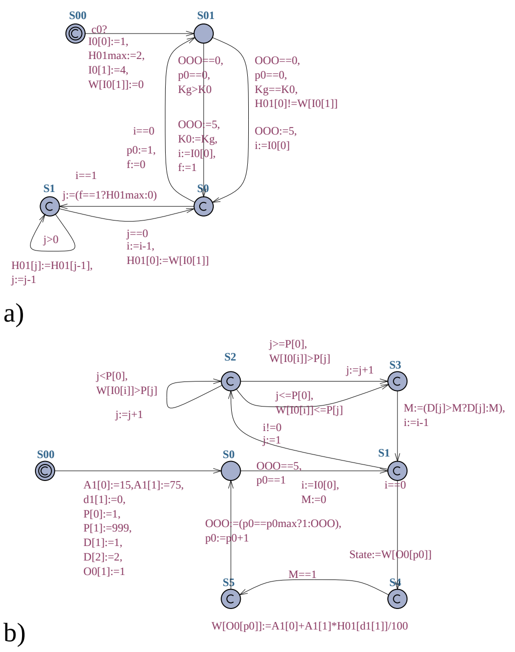

Figure 2. InputsOf (a) and ModeBy (b) automata of a component.

Statespace and intercomponent links. The statespace of a CCP component consists only of its input and output variables. In TA representation, the state of the component is modeled by the global vector $\mathbf{W}$, where values of state variables are stored. The InputsOf type automaton has local variables, referring to the actual input values and ModeBy type automata have variables referring to output values in the vector $\mathbf{W}$.

Transfer function. The transfer function corresponding to the current mode of the component is selected and calculated by ModeBy automata. Selection of the state of the component is modeled by the fragment consisting of automata states S_1, S_2, and S_3 (Figure

2b) and the transitions connecting them. The index of the seleced mode is stored in variable M and appropriate to that index transition $S_4 \longrightarrow S_5$ is taken, which computes the corresponding transfer function.

Latency in transfer functions. The component's transfer function may depend on earlier input values, i.e., some of its delay parameter may be nonzero. That means that in TA representation the values of input I_i have to be stored in the buffer $\mathbf{H}_i$ with length of maximal delay. Having discrete time representation we index time instants by nonnegative integers and input values in $\mathbf{H}_i$ are indexed incrementally beginning from current time instant and increasing to the past, e.g., the latest value is stored in $\mathbf{H}_i(0)$, previous in $\mathbf{H}_i(1)$, etc.

Sceduling in CCP compositions. For proving design correctness it is common to use assume-guarantee type proofs.

When designing the control system we first make assumptions about the behaviour of the plant to be controlled and the environment surrounding it. After that we design the control system, and then attempt to verify that the design satisfies the requirements under the given assumptions. Similarily, each subsystem of the control system can be designed separately and verified with some assumptions of its local environment. It always leads to the separation of two parties - the system components and the environment components, where their interactions define the observable behaviour to be verified. The interaction pattern of the given parties is the following. Within each time instant (global discrete time is assumed) the Environment changes its state generating a stimulus to the System. System reacts to the stimulus computing its own state change. Employing "slow enough system"-assumption we assume that time progresses only when both parties have reached their fixpoints. Unfortunately, this assumption may easily lead to violation of non-Zenoness of computations if those fixpoints are missing. Generally, computations satisfying such modeling assumptions are described by the regular expression $(Environment; System; Clock)^*$. Having even a simple parallel composition of CCP automata this behaviour cannot be guaranteed. We have to introduce a scheduling policy which provides right succession of events. The automata template Scheduler implements this scheduling policy giving the execution right alternatively to System components and Environment components (see Figure 3). The execution right is passed to the next party only after the former has reached its fixpoint. Time is enabled to progress only after System components have terminated. In this solution components within system components and environment components do not have priorities, which means that the component to be executed next is selected nondeterministically from those, the input values of which have been changed but their outputs have not been computed yet. Internally within a component the execution order of transitions is completely deterministic, thus rendering to achieve partial order reduction like effect by simple modeling constraints.

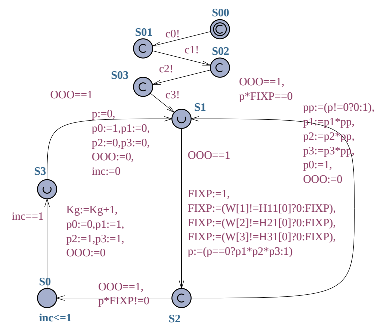

Figure 3. The automaton for sceduling.

2.5 MC query templates

Correctness requirements to be model checked are generally combinations of safety and liveness properties. Safety properties say that some state property P is never violated by any reachable trajectory. In control applications for different reasons we often have to take into account some transfer phenomena, e.g., oscillations, start up, etc. in case of which the safety properties are temporarily allowed to be violated. So we propose conditional safety property saying "safety property P is never violated provided condition C holds". Using timed temporal logic TLTL conditional safety properties can be specified by formula templates $A\square(C \Rightarrow P)$, where A and $\square$ denote resp. modalities "always" and "globally". For instance, to specify that after some transfer time τ the model parameter x always remains within interval $[l; u]$ we have

$$A\square(Time \geq \tau \Rightarrow l \leq x \wedge x \leq u),$$

where $Time$ is a variable modeling discrete time.

Liveness properties express eventualities. Typical question to be asked in RT system design is an upper bound estimate of reaction time. This is an example of bounded liveness property, that can be paraphrased "given a stimulus s at time instant t_s the reaction has to be always not later than at $t_s + d$". To state it formally we need a location variable l_1 and a clock "Clock" that are reset to s and 0 respectively, when the stimulus transition with destination state s is taken in the model. We also need another location variable l evaluated by the current state and a dedicated state r modeling the completion of the reaction event. Then the query template is following:

$$A\diamond(l_1 = s \wedge Clock \leq d \wedge (l = r)),$$

where $\diamond$ denotes the modality "eventually".

3. USING PATTERNS IN MC

Although there are powerful model checking techniques, e.g., symbolic representation, partial order reduction etc. that allow to increase substantially the size of the systems that can be verified, many industrial applications are too large to be handled. The techniques that can be used together with the symbolic methods are compositional reasoning, symmetry, external constraints etc (Clarke *et al.*, 1999). In this section we discuss how these techniques are supported by modeling patterns approach.

Compositionality. Many systems are composed of multiple entities running in parallel. The specification of such systems can often be decomposed into properties that describe the behaviour of each component separately. The pattern approach rests on the idea of modularity and allows to provide the alphabet of subcomponents to decompose a complex system in a systematic way. Having the candidate components the strategy is to check each of the local properties, using only the component defined by selected pattern. If the conjunction of the local properties implies the overall specification, then the complete system satisfies this specification as well.

Symmetry. Systems often contain replicated components. For example, in CCP, automata being instances of InputsOf and ModeBy templates, replicate in each component. This fact is to obtain reduced models for the system. The groups of such replicas can be used to define an equivalence relation on the state space of the system and to reduce the search space.

Extra constraints. Similarly to the partial order reduction the sets of observationally equivalent behaviours can be reduced and uninteresting cases can be eliminated by posing extra constraints to the model. In terms of modeling patterns it means, e.g., restricting the set of pattern behaviours with certain scheduling discipline. Choosing among disciplines such as maximal parallelism, bounded fairness, fixed priorities etc. allows to decrease the number of interleaved state sequences that must be explored in model checking. An implementation of the scheduler imposing bounded fairness discipline is presented in (Vain and Kyttner, 2001). In the current approach we have combined two strategies - fixed priority scheduling on component automata level and fair scheduling on system level where CCP components are competing for the execution right.

4. IMPLEMENTATION

The current implementation of CCP is based on XSLT tranformation. The component model is defined in XML using a DTD (lightweight grammar) based on the component described in Figure 1. The component model is then transformed into the automata network of the Uppaal tool (Larsen and Petterson, 1997) using an XSLT stylesheet and custom extensions written in Java.

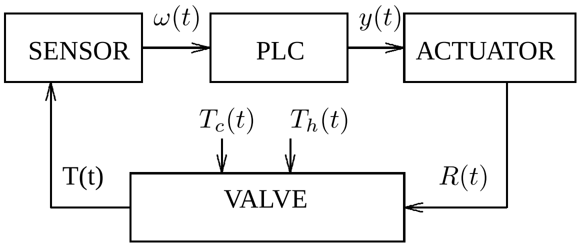

Figure 4. Temperature control system.

The transformation creates the following items:

(1) Static global variables;
(2) Global variables and channels (used by automata corresponding to each component);
(3) The InputsOf and ModeBy automata (see section 2.4 for definition) for each component together with local variables needed by those;
(4) The Scheduler automaton;
(5) The system definition.

The InputsOf and ModeBy automata corresponding to each component are defined using the transformation in favour of Uppaal templates because their structure is parametrized depending on the number of outputs and mode definitions in the component model.

5. CASE STUDY: WATER TEMPERATURE CONTROL SYSTEM

A water temperature control system inspired by (Davidson, 1990) is considered in the case study. Plant to be controlled is a water supply system for a drum boiler where the goal of the control is to keep the temperature of water flowing into the boiler within the range of 70 $\pm$ 4 C°. The regulator has two incoming water pipes - one for hot water and the other for cold water. The control system (Figure 4) consists of a temperature sensor measuring the temperature of the water running into the boiler; programmable logic controller (PLC) and an actuator which is a motor driven hot and cold water mixing valve. It is assumed that the standard analysis for control design is done and transfer functions of components are identified.

Requirements and assumptions. User requirements say that the control system must keep the temperature of the water running into the boiler always within the range 66 - 74 C°. To have a feasible design also some engineering constraints must be met. For instance, due to limited number of allowed switching cycles of the valve motor it is important to avoid over-regulation and to minimize the motor use. A design constraint to controller is introduced saying "if T_{cold} and T_{hot} have been stable within their Reliable regions at least

$t_{stable} = 2$ sec then control activities must be completed and the motor switched off". Characteristics of the plant are the temperatures of incoming hot water and cold water (T_h and T_c, respectively) and the temperature T of mixed water flowing into the boiler. It is assumed that $T_h = 90C^\circ$ and $T_c = 20C^\circ$. The pressure in both inflow pipes is constant and balanced.

Components. Temperature sensor transforms the measured temperature T to frequency ω.

$$\omega(t) = 25 \cdot T(t - \tau_s) + 2830,$$

where $\tau_s = 0.5sec$. Frequency 4630 Hz corresponds to 72 C° and 4530 Hz to 68 C°. Based on sensor output readings the PLC computes in each 0.1 sec a new control signal and switches "ON" one of two output relays providing corresponding PLC output values 1 or -1. Values -1 and 1 denote current with different polarity driving the valve motor in opposite directions. Value 0 denotes the situation where both output relays are "OFF" and the motor stops. The transfer function of the PLC is

$$y(t) = \begin{cases} 1 & 4630 < \omega(t - \tau_c) \\ 0 & 4530 < \omega(t - \tau_c) < 4630 \\ -1 & \omega(t - \tau_c) < 4530 \end{cases}$$

with $\tau_c = 0.1$ sec. Maximum time of turning the valve's position by motor between two extreme positions is 14 sec. The first derivative of valve's relative position $R \in$ [0,1] is $R(t) = k \cdot y(t)$, where $k = 1/14sec^{-1}$. For simplicity it is assumed that mixing cold and hot water in the valve is instantaneous and temperatures are related as in the following equation:

$$T(t) = R(t) \cdot T_h(t) + (1 - R(t)) \cdot T_c(t).$$

Given specification is formalised using above described transformation steps and resultant system consists of the set of automata expressed by the following regular expression:

$$System =$$
$$Scheduler, Timestop, (InputsOf, (ModeBy)^{p_c})^n,$$

where $Timestop$ is an automaton containing a single location and a single clock that restricts the amount of time the system is allowed to run, n is the number of components, and p_c is the number of modes relevant to a particular component c.

Verification. For verification of design correctness it suffices to check the safety formula $A\Box(time > transfer\_time \Rightarrow L\_bound < temp \wedge temp < U\_bound)$, saying that after transfer period the water temperature is always within the range $[L\_bound; U\_bound]$.

6. CONCLUSION

According to the authors' knowledge this is one of the first attempts to combine two powerful techniques - model checking and pattern based design to address the complexity of industrial size design problems. Our work is based on the belief that modeling patterns support scaling up MC tasks otherwise unsolvable on unstructured models. We proposed a Control Component (proto-)Pattern, which regardless of its mathematical simplicity allows to model rather broad class of control system components and their compositions preserving efficient decidability of MC tasks. We have defined the translation of this pattern into a network of timed automata, which opens the way for easier construction of structurally uniform models and for automated checking of these models. This is work in progress and some complexity problems related to reaching fixpoints in feedback model architectures are yet unsolved. Another effort planned as continuation of this work is enabling interchange between CCP and a certain subset of Simulink models. The first step towards simplifying this kind of interchange has been made by applying XML technology in the implementation.

REFERENCES

Clarke, Edmund M., Jr., Orna Grumberg and Lucent Technologies (1999). *Model Checking*. MIT Press. Cambridge, Massachusetts.

Davidson, E. (1990). Benchmark problems for control system design. Report of IFAC Theory Committee. 4-5.

Gamma, Erich, Richard Helm, Ralph Johnson and John Vlissides (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley professional computing. Addison Wesley Publishing Company. Reading, Massachusetts.

Hune, Thomas S., Kim G. Larsen and Paul Pettersson (2000). Guided synthesis of control programs for a batch plant using UPPAAL. Research Series RS-00-37. BRICS. Department of Computer Science, University of Aarhus.

Koutsoukos, Xenofon D., Panos J. Ansaklis, James A. Stiver and Michael D. Lemmon (2000). Supervisory control of hybrid systems. *Proc. of the IEEE* **88**(7), 1026–1049.

Larsen, Kim G. and Paul Petterson (1997). Uppaal in a nutshell. *Springer International Journal of Software Tools for Technology Transfer*.

Vain, Jüri and Marko Kääramees (2000). Towards unified compositional design of control systems. *6th IFAC Workshop on Algorithms and Architectures for Real-Time Control* pp. 41–46.

Vain, Jüri and Rein Kyttner (2001). Model checking - a new challenge for design of complex computer-controlled systems. Proc. of 5th International Conference on Engineering Design and Automation 'Design and Manufacturing Automation for the 21th Century'.