Goal-oriented Specification of Adaptation Requirements in Adaptive Systems

Greg Brown, Betty H.C. Cheng, Heather Goldsby, Ji Zhang
Software Engineering and Network Systems Laboratory
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
{browngr3, chengb, goldsbyh, zhangji9}@cse.msu.edu

1. INTRODUCTION

Increasingly, autonomic systems are being used by diverse users, such as the medical community [16], software industry [4], and in response to chemical or biological terror attacks [1] because of the self-management, self-protection, and self-healing benefits these systems can provide [4] in response to changes in the physical and software environments [11]. Unfortunately, these adaptive systems are very complex and the requirements for such systems may not be fully understood by their designers. This complexity coupled with this lack of understanding may lead to adaptive systems with errant behavior [15], which can offset the potential benefits of a adaptive system. The ability to reason about an adaptive system’s adaptation semantics, that describe how the adaptive software changes from the source program to the target program is crucial. This paper introduces the goal-oriented specifications of three adaptation semantics commonly found in adaptive systems [24], built with the KAOS methodology [6]. Developers may expand models built using this approach to specify the adaptive requirements for their adaptive system.

KAOS has been previously used in the context of adaptive systems. Informally, van Lamsweerde et al. [8] discussed adaptation semantics in adaptive systems in the context of the FLEA runtime event monitoring system [5], where they implicitly specified adaptation semantics with event-condition-action rules. Their approach focused on monitoring requirements conditions that trigger adaptation.

Our approach differs from theirs in that we use KAOS goal models to explicitly specify the adaptation semantics previously introduced by Zhang and Cheng [24], where they introduced an extension to LTL called Adapt operator-extended LTL (A-LTL) and used it to formally specify the semantics they introduced. They extended LTL with the adapt operator (“”) in order to specify adaptation behavior. Generally, the adapt operator is used as \( A \overset{\Omega}{\Rightarrow} B \), indicating that the adaptive software initially satisfies \( A \) and in a later state, it stops being obligated to satisfy \( A \) and starts to satisfy \( B \). The A-LTL specifications of the adaptation semantics can be used in the specification of adaptive systems and can be checked for consistency, correctness, dynamic insertion of
adaptive code into an adaptive system, etc. The original A-LTL specifications are useful, but may not be easy to comprehend at first glance. We encapsulate the A-LTL specifications with KAOS specifications that are more intuitive to understand, but are still amenable to verification and consistency checking. Additionally, the KAOS specification can be refined by the system designer into a complete specification of the adaptive system.

The KAOS methodology provides a graphical way to present the adaptation semantics. The A-LTL semantics specifications were essentially converted into goal-oriented models. We identified the high level objectives of each of the adaptation semantics and represented them as the corresponding KAOS goal entity. Then we identified distinct states present in the adaptive semantics and represented them as KAOS requirements. Next, we identified operations, which describe how state changes occur in the adaptive system and assigned them to requirements to represent the A-LTL adaptation semantics. Finally, we used the A-LTL specifications to provide the formal definitions and pre- and post-conditions of the graphical KAOS elements.

To illustrate this approach, we modeled the adaptation semantics for the adaptive MetaSockets [14] for adaptive mobile computing systems. Section 2 discusses the features of the KAOS specification language pertinent to this paper. Section 3 discusses the KAOS models and A-LTL specifications for the adaptation semantics. Section 5 summarizes the work and discusses future investigations.

2. KAOS SPECIFICATION LANGUAGE

In the following, we give a brief overview of the KAOS goal-oriented specification language and describe in more detail the portion of the language we use in this paper. Further detail about KAOS can be found in [6, 10]. A key for the elements of a KAOS goal model is presented in Figure 1.

![KAOS elements](image)

**Figure 1: The KAOS elements used in this paper**

The KAOS language defines four primary types of models: goal models, agent models, operation models, and object models. A goal model specifies goals, which are stakeholder objectives that the system should achieve. An agent model assigns goals to agents, where an agent is a system component, e.g., an automated component or a human. The agent is responsible for ensuring the achievement of their goals. This model can be inferred from the goal model. An operation model specifies the operations that the agents must perform to achieve the goals expressed in the goal model. An object model identifies the objects that are used in the KAOS model. An object may be either an entity, an agent, or a relationship between objects. An entity is an inanimate object, such as an unplugged piece of hardware, that is unable to perform operations. Next, we provide additional details about goal models, since it is the primary specification language used in this paper.1

Graphically, a goal is depicted as a parallelogram. A goal may be refined into sub-goals and/or requirements that elaborate how the goal is achieved. A goal is AND-refined if its sub-goals must all be achieved for the goal itself to be achieved. As far as this paper is concerned, a goal is OR-refined if its sub-goals describe alternative ways to achieve the goal; OR-refinement may be used different in other notations[20]. Graphically, we represent refinement with a lightly shaded circle that points to the goal it refines. AND-refinements have lines from each sub-goal to the same lightly shaded circle. OR-refinements have lines from sub-goals to different slightly shaded circles.

An agent is graphically represented as a hexagon. We indicate an agent is responsible for a goal or requirement by having an arrow point from the agent to it.

A requirement is a goal under the responsibility of an automated component. Graphically, a requirement is depicted as a parallelogram with a thicker black border.

An operation is an action performed by an agent to achieve a goal by operationalizing a requirement. Graphically, operations are represented as ovals. Operationalization is shown graphically with a dark circle that points to the requirement being operationalized. An agent is responsible for an operation and this responsibility is shown with a circle with lines that points to the operation for which the agent is responsible.

A message is a communication between agents. It is represented graphically as a long, short pentagon with a point towards the right.

A KAOS entity is formally described by several fields. The first field is the type of entity and name of the particular instance. The options that are used in this paper are Goal, Requirement, and Operation. The next field in a Goal or Requirement is the Concerns field, that is used to list the concepts that the entity uses, or is concerned with. The RefinedTo field lists the goals and/or requirements that are used to refine the goal. Next, the Refines field lists the goal that is refined by the entity being described. The InformalDef field gives the informal definition of the entity that is being described. The FormalDef field contains the formal LTL or A-LTL specification of the entity being described.

The Operation specification may have different fields in their descriptions, other than those used to specify goals and requirements. The Input field describes on what type(s) of entities the operation acts. The DomPre field formally

1We follow the color conventions set forth by Cediti for the Objectiver tool [3].
describes the state of the system before the operation is performed. The DomPost field formally describes the state of the system after the operation is performed.

3. ADAPTATION SEMANTICS

In this section, we briefly introduce the example that serves as our case study and then illustrate our approach by specifying the three adaptation semantics using KAOS. We use the same font to denote KAOS model elements.

3.1 Adaptive Example

Zhang and Cheng [24] examined the semantics of adaptive systems in the context of a program that utilizes MetaSockets and performs DES 64 and DES 128 encoding/decoding. A MetaSocket is a special type of socket that is created from an existing Java socket class, but its structure and behavior can vary through adaptation in response to external stimuli at runtime [14]. A KAOS model representation of the MetaSockets program specification is shown in Figure 2.

The goal at the root of the tree, Achieve[MetaSocketSpec], is the overall objective of the MetaSockets adaptive system. The Achieve[MetaSocketSpec] goal is AND-refined by the goals:

Achieve[Insecure output never produced],
Achieve[All input packets are output],
Achieve[No packet output before corresponding input arrives], and
Achieve[AdaptVar].

The Achieve[AdaptVar] goal encapsulates the adaptation functionality of the MetaSocket system. These are all high-level goals that represent some functional requirements of the sample MetaSocket system. In the following, we examine the Achieve[AdaptVar] goal, which encapsulates the adaptive requirements of the MetaSocket system, in greater detail. The entities from the KAOS model are presented in sans serif in the text. Fields of the formal specification are presented in bold.

The Achieve[AdaptVar] goal is further OR-refined in Figure 3. The goal is OR-refined into two complementary goals. The Achieve[Adapt_from_DES_64_to_DES_128] goal is further explored in this paper. The other goal would be refined similarly.

The AND refinement in KAOS does not imply any sort of ordering or serialization preference between the goals in the AND refinement. In many systems, this is not a problem. In the cases where ordering is important, annotation from Yu, et al. [21], has been adopted to clarify the sequence. According to Yu, et al. [21], AND refinements can be annotated as sequential (\(;\)). Sequential annotation is used in this paper and indicates a left to right sequence.

![Figure 3: The OR-refinement for the AdaptVar goal](image)

3.2 Adaptation Semantics Examples

Zhang and Cheng [24] identified three adaptation semantics: one-point, guided, and overlap. In the following, we define and model each adaptation semantic for the DES 64 to DES 128 adaptation using the KAOS notation.

3.2.1 One-point Adaptation

A one-point adaptation is characterized by the system adapting from the source program to the target program at a specific point in execution [24]. Specifically, the system adapts after it receives an adaptation request. The KAOS model for specifying the DES 64 to DES 128 adaptation using one-point adaptation semantics is depicted in Figure 4. The goal, Achieve[Adapt_from_DES_64_to_DES_128], corresponds to the goal of the same name depicted in Figure 3. It is AND-refined into two requirements that embody the states that the system transitions through as it adapts from the source to the target programs. The Change Agent is responsible for triggering the adaptation mechanism that transitions the system between these states. The Monitor Agent monitors the system for conditions that warrant adaptation and then sends a message to the Change Agent. This monitoring is not explicitly shown in the figures in this paper.

![Figure 4: KAOS model of one-point adaptation semantics](image)
Achieve[Adapt_from_DES_64_to_DES_128]. The goal is concerned with the specification of the source and target programs, \( S_{SPEC} \) and \( T_{SPEC} \) respectively, as well as the request for adaptation, \( A_{REQ} \). The source and target programs are elements of adaptability in the overall adaptive system. In the one-point adaptation semantics, \( S_{SPEC} = DES64_{SPEC}, T_{SPEC} = DES128_{SPEC} \), and \( A_{REQ} = Request_64_{to_128}_{OnepointREQ} \).

![Image](54x282 to 293x324)

3.2.2 Guided Adaptation

A guided adaptation is characterized by the system restricting the source program and retaining its program behavior before a safe state is reached [24]. The system then converts via an adaptation to the target program. The KAOS model for specifying the DES64 to DES128 adaptation using guided adaptation semantics is depicted in Figure 5. Similar to the model of one-point adaptation semantics (depicted in Figure 4), the overall goal, Achieve[Adapt_from_DES_64_to_DES_128], corresponds to the goal of the same name depicted in Figure 3. It is refined to Achieve[Adapt_from_DES_64_Restricted_to_DES_128] goal, which is specific to the guided adaptation semantics. This goal is further AND-refined into a goal and a requirement, annotated as sequential (\( \cdot \)). We have a goal and a requirement in this case because the guided adaptation semantics has an intermediate state between the two in the one-point semantics which represents restricted behavior of the source program. The goal represents the objective of moving from the source program behavior (i.e., \( in_{DES64_{State}} \)) to the restricted source program behavior (i.e., \( in_{DES64_{Restricted}_{State}} \)). The requirement, \( in_{DES128_{State}} \), represents the state of the system when it exhibits DES128 behavior. The Achieve[Adapt_from_DES_64_to_DES_64Restricted] represents an objective to move from the source program behavior to exhibiting restricted behavior. The goal is still further AND-refined into two other sequentially annotated requirements. These three requirements embody the states that the system has during its transition from the source to the target programs. The Change Agent is responsible for these state changes. It performs a state change based on the messages it sees, which could be either the \( A{Request_64_{to_64}}_{REQ} \) message or the \( A{Request_64_{to_128}}_{REQ} \) message, meaning to carry out the first part of the semantics from the source program to the restricted source program and to carry out the second part of the semantics from the restricted source program to the target program, respectively. Both messages occur when there is an adaptation request made of the Change Agent and are constituent parts of the request.

The formal specification of the elements depicted in this model is as follows:

![Image](54x334 to 293x394)

![Image](54x547 to 293x637)
The first goal is the same as the one for one-point adaptation. It is a high-level goal having the system adapt from its DES 64 program behavior to its DES 128 program behavior.

We refine the above goal to obtain the subgoal Achieve[Adapt_from_DES_64_Restricted_to_DES_128], that includes the restriction condition \( R_{COND} \), which must be satisfied before adaption to DES 128 behavior. Specifically, \( R_{COND} \) is used to specify the safe states in which the system can adapt [23] without putting the system into an inconsistent state.

![Figure 5: KAOS model of guided adaptation semantics](image)

The requirements in Figure 5 embody the state of the system that describes the three possible conditions for the guided adaptation. The \( \text{In}_{\text{DES}} \) states specify the system when it has received an adaptation request, \( \text{Request}_{64 \to 64} \), and is waiting for a safe state where it can adapt to the target program and stop acting like the source program. The \( \text{In}_{\text{DES}} \) states and \( \text{In}_{\text{DES}} \) states requirement specifications are identical to those in one-point adaptation, in Figure 4, and are not included for space reasons.

Operations similar to those in the one-point semantics are also necessary for communication between the agents in the program to control when the adaptations occur for guided adaptation semantics, as well as for converting between the \( \text{In}_{\text{DES}} \), \( \text{In}_{\text{DES}} \), and \( \text{In}_{\text{DES}} \) states. These are shown in Figure 5.

### 3.2.3 Overlap Adaptation

An overlap adaptation is characterized by an overlap of the source and target programs [24]. The system monitors the input and eventually stops the source program when appropriate, where the target program may have been executing before the source program stops executing. The KAOS model for specifying the DES 64 to DES 128 adaptation using overlap adaptation semantics is depicted in Figure 6. Similar to the model of one-point adaptation semantics (depicted in Figure 4), the overall goal, Achieve[Adapt_from_DES_64_to_DES_128], corresponds to the goal of the same name depicted in Figure 6.

The formal specification of the elements depicted in this model is as follows. The overall goal has slightly different InformalDef and FormalDef fields than those used in the guided adaptation semantics in order to reflect the difference in semantics. In the these specifications, \( R_{COND} \) specifies a restriction condition that is placed on the source program.
This next goal is similar to the analogous goal from the guided semantics, but the InformalDef and FormalDef are different to reflect the different semantics. The goal represents the adaptive system changing from the source program, DES 64, to the parallel execution of the source program with the target program DES 128.

The In DES 64 State and In DES 128 State requirements in Figure 6 are identical to those of the one-point adaptation semantics shown in Figure 4.

3.3 Discussion

The KAOS models of the adaptation semantics have advantages over the original A-LTL specifications. The KAOS models for the adaptation semantics have a general graphical form. The models for the guided and overlap adaptations are interchangeable and the one-point adaptation appears to be a special case of that model. The primary difference between them is in the A-LTL descriptions used in the formal definitions of the KAOS elements.

The KAOS specifications can also be used as the basis of goal-oriented specifications of adaptive systems. With the graphical presentation of the adaptive semantics, it may be more intuitive for adaptation developers to identify patterns of adaptive behavior across a single adaptive system or across a collection of adaptive systems. Furthermore, the goal specifications of the adaptation semantics can be integrated seamlessly to the other goals of the system.

4. RELATED WORK

There are two notable goal-oriented approaches to specifying adaptive systems. Specifically, Feather et al. [8] used KAOS to specify adaptive system requirements. Their approach also included inferring runtime monitors of system requirements from the KAOS specifications. Lapouchnian used Tropos/i* to model user goals [21] and to configure personal software [22]. Lapouchnian et al. [20] also specified system-wide adaptation goals using a hybrid goal-oriented
modeling technique that includes concepts and notations from both KAOS and i* [19]. Their approach defines the system features, possible adaptations, and determines when to adapt; whereas, our approach specifies what is expected to occur during the adaptation.

The KAOS methodology has been used for several industrial systems. Specifically, van Lamsweerde et al. [17] used the KAOS methodology to elaborate the requirements of a meeting scheduler and introduced refinement patterns in the context of KAOS [7]. These patterns were intended to capture commonly occurring situations when modeling software.

There are also alternative goal-oriented requirements modeling languages to KAOS for modeling adaptation concerns. Mylopoulos et al. [2, 12] introduced an agent-oriented modeling methodology called Tropos, which uses i* [19] for requirements-driven software development [2, 12]. Similar to KAOS, Tropos/i* use graphical representations of software concepts. In other work, Mylopoulos et al. modeled goals and requirements of an online media store [2] and Perini et al. [13] used Tropos/i* to produce models of a software system. Fuxman et al. [9] discussed model checking of early requirements specifications in Tropos/i*.

5. SUMMARY AND FUTURE WORK
In summary, this paper presents an approach for specifying three adaptation semantics identified by Zhang and Cheng using the KAOS methodology. This approach provides the adaptation system developer with two significant benefits.

First, the use of the KAOS methodology allows the adaptation developer to specify the features and adaptation semantics of an adaptive system in the same model. Thus, the adaptation developer can draw on the extensive research done on goal-oriented specification of system features [6, 18].

Second, the relationship between the adaptation semantics’ KAOS specification and formal A-LTL specification allows the adaptation developer to achieve the benefits of formal specification while using the easier to understand graphical notation. Specifically, some of the formal specification benefits of this approach include verification by consistency checking, correctness checking, and checking the dynamic insertion of adaptive code into an adaptive system.

Our ongoing research includes the adaptation semantics themselves as well as several related issues. These include the decision-making strategies that are used by the system to decide when and how to adapt. Also, the use of these adaptation semantics models in different application domains is a subject of ongoing research.

6. REFERENCES