Stepwise Context Boundary Exploration Using Guide Words

Naoyasu Ubayashi and Yasutaka Kamei

Kyushu University, Japan
ubayashi@acm.org, kamei@ait.kyushu-u.ac.jp

Abstract. Most requirements elicitation methods do not explicitly provide a systematic way for deciding the boundary of the usage context that should be taken into account because it is essentially difficult to decide which context element should be included as the system requirements. If a developer explores the context boundary in an ad-hoc manner, the developer will be faced with the *frame problem* because there are unlimited context elements in the real world where the target system exists. There are many application domains that should take into account the *frame problem*: security, safety, network threats, and user interactions. To deal with this problem, this paper proposes a new type of requirements analysis method for exploring the context boundary using guide words, a set of hint words for finding a context element affecting the system behavior. The target of our method is embedded systems that can be abstracted as a sensor-and-actuator machine exchanging the physical value between a system and its context. In our method, only the *value-context elements*, a kind of *value objects*, are extracted as the associated context elements. By applying the *guide words*, we can explore only a sequence of context elements affecting the data value and avoid falling into the *frame problem* at the requirements analysis phase.

Keywords: Context analysis, Frame problem, Embedded systems.

1 Introduction

Many embedded systems not only affect their context through actuators but also are affected by their context through sensors. The term *context* refers to the real world such as the usage environment that affects the system behavior.

In most cases, context is only roughly analyzed in comparison to functional or non-functional system requirements. As a result, unexpected behavior may emerge in a system if a developer does not recognize any possible conflicting combinations between the system and its context. It is also difficult to decide the boundary of the context that should be taken into account: which context element, an object existing outside of the system, should be included as the targets of requirements analysis.

If a developer explores the context boundary in an ad-hoc manner, he or she will be faced with the *frame problem* [14] because there are unlimited context elements in the real world where the system exists. The *frame problem* is

S. Nurcan (Ed.): CAiSE Forum 2011, LNBIP 107, pp. 218–233, 2012.

the problem of representing the effects of the system behavior in logic without explicitly specifying a large number of conditions not affected by the behavior.

To relax the *frame problem* in embedded systems, we propose CAMEmb (Context Analysis Method for Embedded systems), a context-dependent requirements analysis method. A context model is constructed from the initial system requirements by using the *UML Profile for Context Analysis*. This context model clarifies the relation between a system and its context. In CAMEmb, only the *value-context elements*, a kind of value objects, are extracted as the associated context elements because many embedded systems are abstracted as a sensor-and-actuator machine exchanging the physical value between a system and its context. Applying the *Guide Words for Context Analysis*, we can explore only a sequence of context elements directly or indirectly affecting the data value observed or controlled by the system sensors and actuators. Other context elements not affecting the system observation and control are not taken into account because these context elements do not affect the system behavior. We can relax the *frame problem* because we only have to consider limited number of context elements as the context of the target system.

The remainder of this paper is structured as follows. In Section 2, problems in the current requirements analysis methods are pointed out in terms of the *frame problem*. In Section 3 and 4, CAMEmb is introduced to relax the *frame problem*. In Section 5, we discuss how to apply our idea to other domains such as security. In Section 6, we introduce related work. Concluding remarks are provided in Section 7.

2 Motivation

In this section, typical problems in the current requirements analysis methods are pointed out by describing the specification of an electric pot as an example.

2.1 Motivating Example

An electric pot is an embedded system for boiling water. Here, for simplicity, only the following is considered: 1) the pot has three hardware components: a heater, a thermostat, and a water level sensor; 2) the pot controls the water temperature by turning on or off the heater; 3) the pot changes its mode from the heating mode to the retaining mode when the temperature becomes 100 Celsius; and 4) the pot observes the volume from the water level sensor that detects whether water is below or above a certain base level.

In case of the electric pot, the water temperature should be taken into account as an important context element. Here, as an example, let us consider the specification that controls the water temperature. In most cases, this specification is described by implicitly taking into account the specific context—for example, such the context that water is boiled under the normal air pressure. A developer describes the software logic corresponding to the specific context—in this case, the pot continues to turn on a heater switch until the water temperature becomes 100 Celsius. Below is the specification described in pseudo code. This

function describes that a controller continues to turn on a heater while the value of the temperature obtained from a thermostat is below 100 Celsius. The `Boil` function behaves correctly under the normal circumstance.

```
// Boil function
while thermostat.GetTemperature() < 100.0
  do heater.On();
```

Although this traditional approach is effective, there is room for improvements because it does not explicitly consider the context elements such as water and air pressure. The above `Boil` specification looks correct. However, faults may occur if the expected context is changed—for example, the circumstance of the low air pressure. Because the boiling point is below 100 Celsius under this circumstance, the software controller continues to heat water even if its temperature becomes the boiling point. As a result, water evaporates and finally its volume will be empty. The water level sensor observes the volume, and the pot stops heating. Although this behavior satisfies the above system specification, the pot may be useless for the people who use it up on high mountains where the air pressure is low.

2.2 Problems to be Tackled

The boundary of the context should be determined from stakeholders' requirements. If we consider climbers as customers of the pot, we have to admit that we failed in eliciting requirements in the above example.

It is not easy to define the context boundary even if the target users of the system are determined. A developer will be faced with the *frame problem* because there are unlimited context elements in the real world. There are some studies that take into account the real world as a modeling target. For example, Greenspan, S. et al. claim the necessity of introducing real world knowledge into requirement specifications [4]. But, current requirements elicitation methods do not answer a question: how and why do we find air pressure as a context element ? Of course, domain knowledge and past experiences are important to find this kind of requirements elicitation. Moreover, we admit that there are no complete methods to overcome the *frame problem*. However, at the same time, we need a method for systematically exploring the context boundary because many incidents that occur in the real embedded systems are caused by insufficient context analysis. That is, unexpected context influence that cannot be predicted in the requirements elicitation phase tends to cause a crucial incident. Many engineers in the industry face this problem.

3 CAMEmb

CAMEmb is a context analysis method for dealing with the problem pointed out in Section 2.

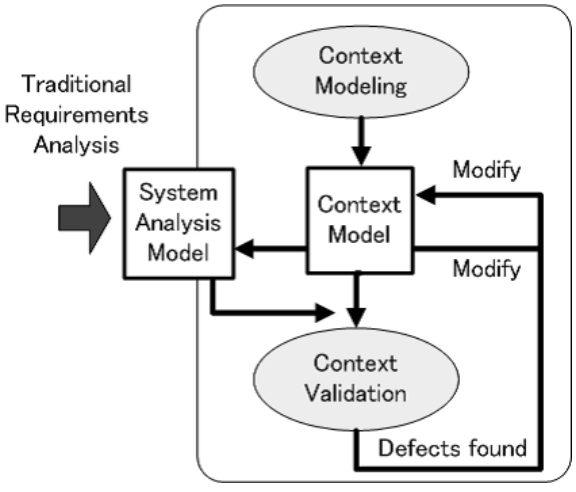

Fig. 1. CAMEmb overview

3.1 Overview

CAMEmb complements the insufficiency of the traditional requirements analysis methods as illustrated in Figure 1.

CAMEmb in the requirements analysis phase consists of 1) context modeling and 2) context validation. After traditional requirements analysis is performed from the viewpoint of eliciting the system functions and non-functional properties, CAMEmb is applied. In 1), the context elements affecting the system behavior are extracted. The boundary of the context that should be taken into account is explored. In 2), the consistency and correctness of a context model is verified using VDM++ [3], an object-oriented extension of VDM-SL (The Vienna Development Method – Specification Language). We can check whether a system analysis model behaves correctly within the expected context boundary. When the system analysis model does not behave correctly, we regard this result as the requirements elicitation defects. The context boundary is not correct or the system requirements are not feasible in the expected context. In the former case, we have to modify the context model. Otherwise, in the latter case, we have to modify the system analysis model. It depends on stakeholders' needs whether we have to modify the system analysis model or the context model.

In this paper, we focus on the context modeling method and explain its process step by step.

3.2 Context Analysis Model

Figure 2 illustrates the result of context analysis for an electric pot. The upper side and the lower side show a system and its context, respectively. The details of the *Controller* in the context model are described in the system analysis model.

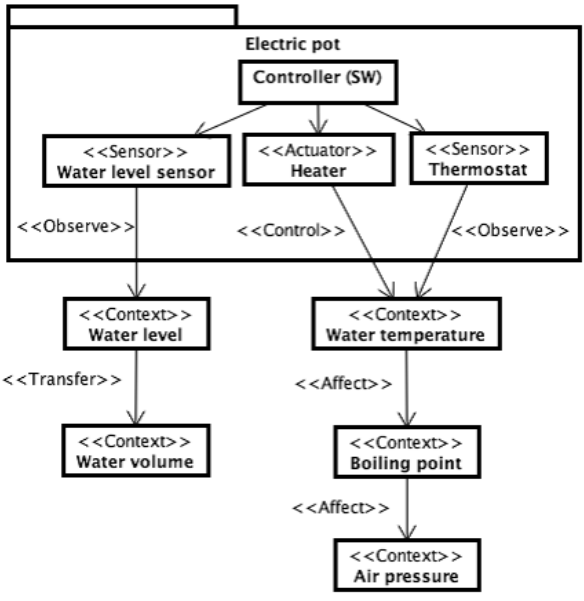

Fig. 2. Context analysis model for an electric pot

Sensors and actuators for observing or controlling the context are regarded as the interface components that separate the context from a system. Figure 2 shows only the structural aspect of the context modeling. The details of the *Controller* and the behavioral aspect of the context model are omitted due to the space limitation. In CAMEmb, the behavioral aspect is modeled using state machine diagrams. The structural aspect plays an important role in exploring the context boundary as mentioned below.

3.3 UML Profile for Context Analysis

A UML profile is provided for context analysis as shown in Table 1.

This profile can describe system elements, context elements, and associations between them: four kinds of stereotypes including $\ll Context \gg$, $\ll Hardware \gg$, $\ll Sensor \gg$, and $\ll Actuator \gg$ are defined as an extension of the UML class ($\ll Sensor \gg$ and $\ll Actuator \gg$ are subtypes of $\ll Hardware \gg$); and five kinds of stereotypes including $\ll Observe \gg$, $\ll Control \gg$, $\ll Transfer \gg$, $\ll Affect \gg$, and $\ll Noise \gg$ are defined as an extension of the UML association. The arrow of $\ll Observe \gg$ and $\ll Control \gg$ indicates the target of observation and control. The arrow of $\ll Noise \gg$ and $\ll Affect \gg$ indicates the source of noise and affect, respectively. The arrow of $\ll Transfer \gg$ indicates the source of transformation. $\ll Transfer \gg$ is introduced in order to represent data transformation because a sensor cannot

Table 1. UML profile for context analysis

Name	Category	Definition
$\ll Context \gg$	Class	Context
$\ll Hardware \gg$	Class	Hardware
$\ll Sensor \gg$	Class	Sensor (subtype of $\ll Hardware \gg$)
$\ll Actuator \gg$	Class	Actuator (subtype of $\ll Hardware \gg$)
$\ll Observe \gg$	Association	Sensor observes a context element
$\ll Control \gg$	Association	Actuator controls a context element
$\ll Transfer \gg$	Association	Data is transformed into a different form because a sensor cannot directly observe the original data
$\ll Affect \gg$	Association	Data from the target context element is affected by other context elements
$\ll Noise \gg$	Association	Noise from other context elements (subtype of $\ll Affect \gg$)

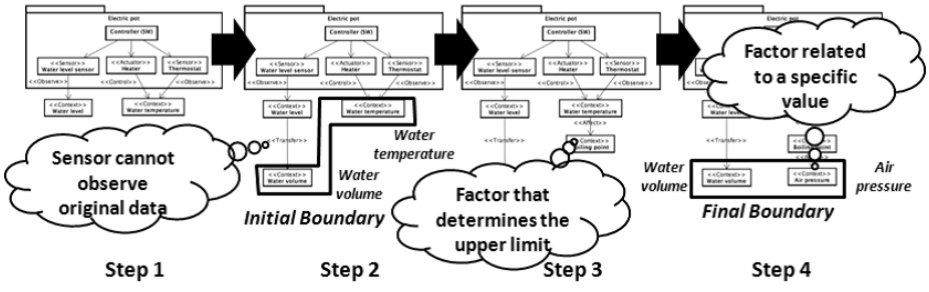

Fig. 3. Stepwise context analysis using guide words (for illustration only)

directly observe the original data. For example, *water level* observed by the water level sensor is not the final observation target of a pot. That is, a pot wants to observe not the *water level* but the *water volume*.

The associations between *Controller* and three hardware components (sensors and actuators) indicate the phenomena such as *sending a command from software to hardware* and *receiving data from hardware*. However, stereotypes for these phenomena are not provided in our UML profile because they should be considered in system analysis not in context analysis.

4 Stepwise Context Analysis Using Guide Words

The context model shown in Figure 2 is created as illustrated in Figure 3. Figure 3 shows only the image of context analysis procedures. Please refer to Figure 2 when a detailed analysis result is needed.

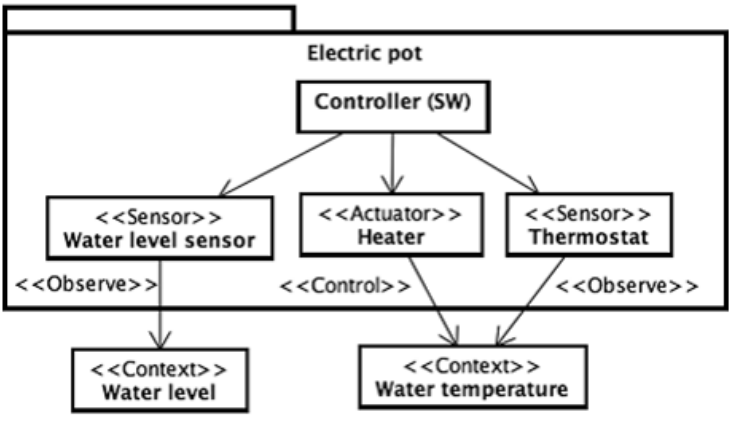

Fig. 4. Step1: extract directly observed or controlled context elements

Step1: Extract Directly Observed or Controlled Context Elements

First, context elements ($\ll Context \gg$), which are directly observed or controlled by a sensor or an actuator, are extracted as illustrated in Figure 4.

We regard the environment value as a context element because CAMEmb focuses on embedded systems based on sensing and actuating. We call these context elements *"value-context elements"*. In case of an electric pot, *water level* and *water temperature* are extracted since *water level* is observed by the water level sensor and *water temperature* is controlled by the heater.

Step 2 [Initial Boundary]: Extract Indirectly Observed or Controlled Context Elements

An element directly observed by a sensor may be an alternative context element in such a case that the sensor cannot observe the original value of the target context element. For example, the pot wants to observe not the *water level* but the *water volume*.

Next, we explore the target context elements by using $\ll Transfer \gg$. In the step 2, all paths from sensors and actuators to the target context elements are completely extracted as illustrated in Figure 5. The initial context boundary is determined in this stage. In case of an electric pot, *water volume* and *water temperature* are extracted as the initial context boundary.

Step 3 [Intermediate Boundary]: Extract Impact Factors Using Guide Words

The initial context boundary is an ideal boundary in which system's sensing and controlling are not affected by other factors. However, there are many factors affecting observation and actuation in the real world. We have to extract these factors in order to develop reliable embedded systems.

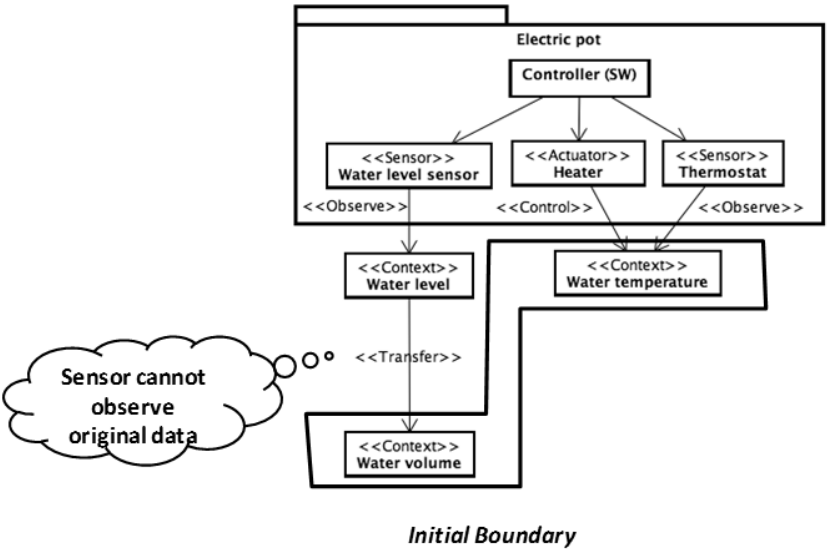

Fig. 5. Step 2 [Initial boundary]: extract indirectly observed or controlled context elements

Table 2. Guide words for context analysis

No.	Category of $\ll Affect \gg$	Guide word
1.	physical phenomena	factor that determines the upper limit
2.	physical phenomena	factor that determines the lower limit
3.	physical phenomena	factor related to a specific value
4.	influence to sensing	factor that interferes with the observation
5.	influence to actuation	factor that interferes with the control

In CAMEmb, impact factors that affect the states of these context elements are extracted using guide words. Guide words, hints for deriving related elements, are effective for software deviation analysis [13]. Guide words are mainly used in HAZOP (Hazard and Operability Studies). In HAZOP, deviation analysis is performed by using the guide words including *NOT*, *MORE*, *LESS*, *AS WELL AS*, *PART OF*, *REVERSE*, and *OTHER THAN*. For example, *higher pressure*, which may be deviated from a normal situation, can be derived from the property *pressure* and the guide word *high*.

In addition to the HAZOP guide words, CAMEmb provides a set of guide words specific to the context analysis as shown in Table 2. These guide words help us to find an obstacle that affects the system observation and control in terms of the *context-value*. By using these guide words, we can extract context elements that affect the context elements existing within the initial boundary. Our guide words can be considered as hints for deviation analysis targeted to context analysis. If there is a context element having the influence on another context element, we link them by the $\ll Affect \gg$ association.

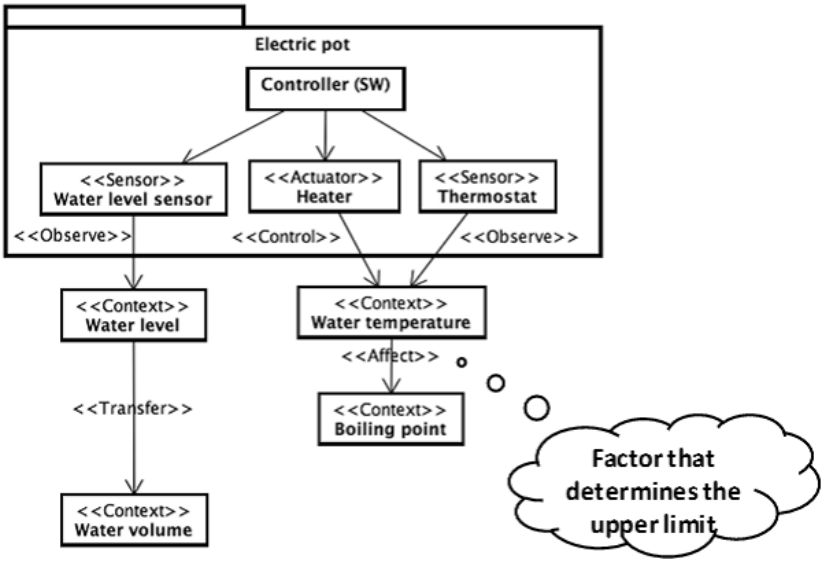

Fig. 6. Step 3 [intermediate boundary]: extract impact factors using guide words

In case of an electric pot, the *boiling point* can be extracted as an impact factor for the *water temperature* by applying the guide word *"factor that determines the upper limit"* since the temperature does not become higher than the boiling point. This guide word indicates that we have to take into account the *boiling point* when we develop an embedded product controlling the *water temperature*. Figure 6 shows this stage of the context analysis. Without guide words, we have to explore impact factors in an ad-hoc manner and we may not be able to find any impact factors.

Step 4 [Final Boundary]: Determine the Context Boundary

We have to continue to extract impact factors as many as possible to develop reliable systems. In case of an electric pot, the *air pressure* can be extracted as an impact factor for the *boiling point* by applying the guide word *"factor related to a specific value"* since the boiling point of the water is 100 Celsius under the circumstance of 1.0 atm. At this point, we finish the context exploration because we can find no more impact factors affecting the *air pressure*. Figure 7 shows this stage of the context analysis. We can extract two context elements *water volume* and *air pressure* as the final context boundary.

It depends on the domain knowledge or experience of a development organization when a developer stops exploring related value-contexts elements. Our method helps a developer to extract context elements affecting the system behavior as many as possible using domain knowledge inspired by guide words. Without domain knowledge, it is not necessarily easy to find impact factors only using guide words. For example, a developer, who does not have enough knowledge about

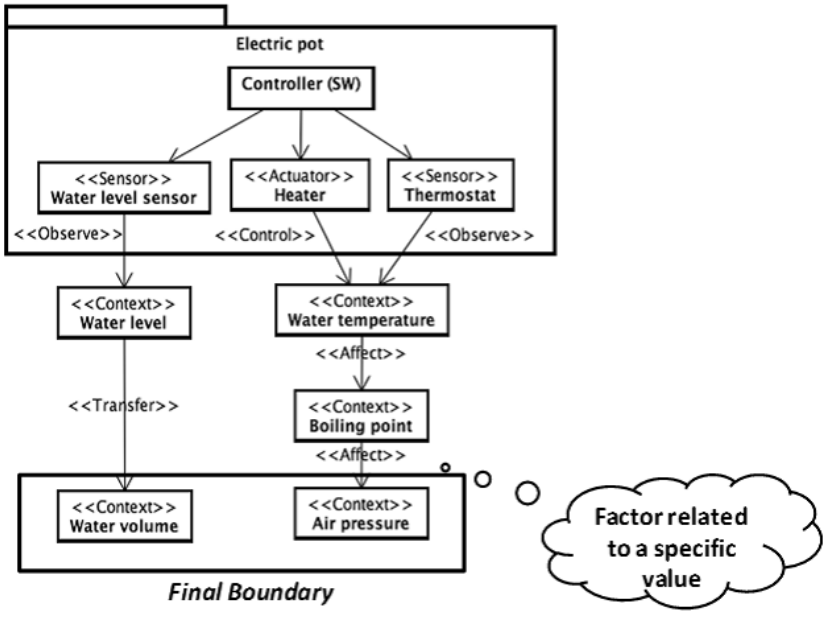

Fig. 7. Step 4 [Final boundary]: determine the context boundary

physics, may not be able to find the *air pressure* even if the developer knows the guide word *"factor related to a specific value"*. Domain knowledge plays an important role in context modeling. However, it is insufficient unless a systematic way for extracting domain knowledge exists. Our guide words can derive domain knowledge needed for context-based deviation analysis from a developer.

As shown here, the boundary of the context is explored by using *UML Profile for Context Analysis* and *Guide Words for Context Analysis*. We can explore only a sequence of context elements directly or indirectly affecting the data value observed or controlled by the system sensors and actuators. Other context elements not affecting the system observation and control are not extracted. There are many context elements such as person, table, and light in the environment of an electric pot. However, these context elements do not affect the data observed or controlled by the pot. So, we do not have to take into account these context elements. These context elements exist out of the boundary.

5 Discussion

In this section, we discuss on the applicability of CAMEmb.

5.1 Applicability

We examined the proposed method to another embedded system: a line trace car. The car runs tracing a line by observing a ground color.

Table 3. Applicability of guide words

No.	Category of $\ll Affect \gg$	Guide word	Example	Case study
1.	physical phenomena	factor that determines the upper limit	boiling point	electric pot
2.	physical phenomena	factor that determines the lower limit	freezing point	electric pot
3.	physical phenomena	factor related to a specific value	air pressure	electric pot
4.	influence to sensing	factor that interferes with the observation	light	line trace car
5.	influence to actuation	factor that interferes with the control	light	line trace car

In CAMEmb, *guide words* play a important role to find context elements affecting the system behavior. Table 3 shows how guide words are applied to two case studies: electric pot and line trace car. Although both of two case studies are sensor-actuator systems, their characteristics are different. In an electric pot, its system behavior is affected by physical environment in which the pot exists. On the other hand, in a line trace car, its system behavior is affected by obstacles of sensing and actuating. There are three kinds of $\ll Affect \gg$: *physical phenomena*, *influence to sensing*, and *influence to actuation*. The guide words related to *physical phenomena* are used in the electric pot. The guide words related to *influence to sensing* and *influence to actuation* are used in the line trace car.

As mentioned here, we can apply our approach to two kinds of sensor-actuator systems.

Although we may not be able to apply CAMEmb to all the application domains, there are many domains that can be modelled as monitor-controller (or sensor-actuator) systems. Security, safety, network threats, and user interactions are examples of such domains. In these domains, context can be analyzed using our approach. For example, *trust* in the security domains correspond to *value* in CAMEmb. By defining the *guide words* that affect the trusts, we can explore the trust boundary.

5.2 Avoidance of the Frame Problem

In CAMEmb, we select only the elements affecting the data value observed or controlled by a system. We think that the value-based context analysis is reasonable because most embedded systems observe the input data from the environment through sensors and affect the environment by emitting the physical outputs through actuators. The system behavior is determined by the data observed by the sensors and controlled by the actuators. We have only to take into account the context elements explicitly or implicitly affecting the data linked with the $\ll Transfer \gg$ or the $\ll Affect \gg$ associations. The context analysis terminates when there are no more context elements affecting the data. In our approach, the affection is determined by using guide words. Of course, the method using guide words is not complete. But, the method helps a developer to find the context elements affecting the system behavior as many as possible.

6 Related Work

Jackson, M. proposes the *problem frames approach* [9] in which relations between a machine (a system to be developed) and the real world are explicitly described. The *problem frames approach* emphasises on the importance of analysing the real world and the problems.

First, in this section, we discuss on the relation between CAMEmb and the *problem frame approach*. There are several common ideas between them. We believe that CAMEmb provides a fruitful mechanism for using the *problem frames approach* more effectively. The *problem frames approach* is strong in analysing the real world (context) in terms of the problems. On the other hand, CAMEmb is strong in exploring the context boundary and refining a context model to the corresponding software design model.

Next, we show the other related work.

6.1 Problem Frames

A context diagram in the *problem frames approach* describes problem domains in an application domain, their connections, and a machine and its connections to the problem domains. The notion of context in CAMEmb corresponds to the real world in the problem frame. Examples of formalising requirements with problem frames can be found in [2][6].

We are now exploring the possibility of integrating CAMEmb with the *problem frames approach*. Figure 8 shows a context analysis model described in the *problem frames approach*. We can describe the context diagram of a line trace car by using the *Required Behavior* frame and the *Transformation* frame. There is the similarity between our UML profile and frame patterns. For example, $\ll Transfer \gg$ corresponds to *Transformation* frame.

We consider that it is effective to apply CAMEmb after problem analysis is done. The *problem frames approach* is strong in analysing problems in the real world. On the other hand, our approach provides a systematic way for determining the context boundary. Because a context analysis model in the *problem frames approach* and that in CAMEmb can be converted each other as shown in Figure 8, we believe that both methods can be integrated.

6.2 Four-Variable Model

Parnas, D. L. and Madey J. propose the *four-variable model* [15], a model for system requirements and design. This model takes into account environmental quantities such as physical proprieties, values displayed on devices, and states of controlled devices. In the *four-variable model*, functions, timing, and correctness are described by using monitored variables, control variable, and input / output data items because variables internal to the system are not adequate for specifying system requirements. The *four-variable model* was used to specify the requirements for the A-7 aircraft in SCR (Software Cost Reduction) [7][8] providing a tabular notation for specifying requirements.

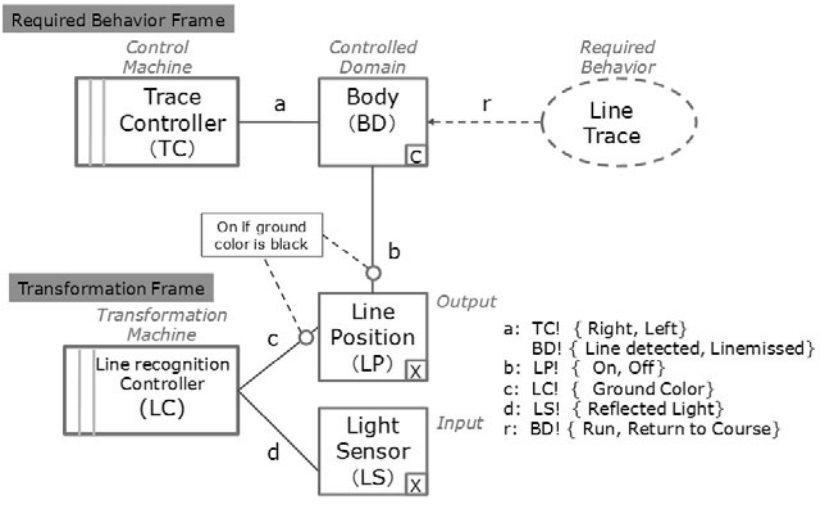

Fig. 8. Context analysis model described in problem frames

The *four-variable model* is similar to CAMEmb because monitored variables and control variables correspond to context elements observed by sensors and controlled by actuators. However, the *four-variable model* does not provide a way for finding context elements and exploring the context boundary. CAMEmb focuses on the recognition paths affecting the system observation and control.

6.3 Context Analysis and SPL

Software product line (SPL) [11] is a promising approach to developing embedded systems. A product is constructed by assembling core assets, components reused in a family of products. In the current SPL, however, the feature analysis is mainly conducted from the viewpoint of system configurations and context is not considered explicitly. However, some PLE methods provide a way for modeling the context of a product [1][16][5][17][12].

Atkinson, C. et al. propose a PLE method called KobrA [1] in which the context realization models are described by analyzing contexts of target systems. However, the systems and contexts are simultaneously described in KobrA. On the other hand, the system lines are completely separated from the context lines in our approach. We believe that our approach is effective comparing to the way in which contexts are taken into account as one of the system concerns. There are contexts features that can be shared among multiple system lines. If a context belongs to a specific system line, the context cannot be reused in other system lines.

Kang, K. C. et al. propose a method for categorizing features into four layer including capability, operating environment, domain technology, and implementation technique [10]. They point out the importance of introducing the viewpoint of *usage context*. Lee, K. and Kang, K. C. propose a model showing how product usage contexts are related to product features [12]. In the model, physical

contexts indicate physical environments or locations where a product is deployed and used.

Reiser, M. O. et al. use the concept of a product set to describe contextual constraints for feature election. Hartman, H. and Trew, T. propose a context variability model (CVM) [5] for modeling product lines that support several dimensions in the context space. Their approach is similar to our approach. In CVM, a separate feature diagram is used to model contextual variability and linked to a product line feature diagram. Similarly, Tun et al. describe contextual variability as a separate feature diagram [17]. The work of Tun is also based on the work of Jackson's problem frames.

We previously proposed a context-dependent SPL method [19] in which a product line was composed of system and context lines. The former is obtained by analyzing a family of systems. The latter is obtained by analyzing the features of the expected context. The configuration of selected system components and context elements can be formally checked by using VDMTools. Historically, the prototype of CAMEmb was originally proposed as a method for constructing context lines. After that, we became aware that our approach could be generalized and could be applied to not only SPL but also requirements elicitation. Moreover, we became aware that the idea of *value-context elements* could play an important role to relax the *frame problem*. In this paper, we positioned CAMEmb as a requirements elicitation method for exploring the context boundary.

7 Conclusion

In this paper, we proposed CAMEmb, a context-dependent requirements analysis method. As demonstrated in this paper, we could provide a method for exploring the context boundary. The idea of *value-context elements* and *guide words* plays an important role.

It is favorable if a system analysis model and a context analysis model are transformed into a design model reflecting the contexts. We have developed a prototype of a domain-specific modeling environment for supporting context analysis [18]. This tool consists of a model editor for supporting a *UML Profile for Context Analysis*, a model compiler for transforming a system analysis model and a context model to a system design model, and a code generator from the system design model to Java. The generated design model consists of three layers *Controller*, *Context Recognition*, and *Driver*. The results of context analysis can be reflected to the software architecture of the target system. The *Context Recognition* layer, the most important part in the design model, is obtained by $\ll Transfer \gg$ and $\ll Affect \gg$ relations in the context model.

We plan to develop a method for integrating CAMEmb with the *problem frames approach*. The former is strong in exploring the context boundary and the latter is strong in analysing the real world. Tool support is necessary to accomplish this research goal. For example, translation between a context model described in CAMEmb and a context diagram in the *problem frames approach* should be supported. It would be better if problem analysis, context analysis, and refinement to a design model are seamlessly linked.

We think that the essential idea of CAMEmb can be applied to other kinds of context such as security and safety in embedded systems. As the next step, we plan to apply CAMEmb to such an application.

Acknowledgement. This research is being conducted as a part of the Grant-in-aid for Scientific Research (B), 23300010, 2011 by the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. Atkinson, C., et al.: Component-Based Product Line Engineering with the UML. Addison-Wesley (2001)
2. Coleman, J.W., Jones, C.B.: Examples of how to Determine the Specifications of Control Systems. In: Proceedings of Workshop on Rigorous Engineering of Fault-Tolerant Systems (REFT 2005), pp. 65–73 (2005)
3. Fitzgerald, J., Larsen, G.P., Mukherjee, P., Plat, N., Verhoef, M.: Validated Designs for Object-oriented Systems. Springer (2005)
4. Greenspan, S., Mylopoulos, J., Borgida, A.: Capturing More World Knowledge in the Requirements Specification. In: Proceedings of International Conference on Software Engineering (ICSE 1982), pp. 225–234 (1982)
5. Hartmann, H., Trew, T.: Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains. In: Proceedings of the 12th International Software Product Line Conference (SPLC 2008), pp. 12–21 (2008)
6. Hayes, I., Jackson, M., Jones, C.: Determining the Specification of a Control System from That of Its Environment. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 154–169. Springer, Heidelberg (2003)
7. Heitmeyer, C.L., Bull, A., Gasarch, C., Labaw, B.G.: SCR*: A Toolset for Specifying and Analyzing Requirements. In: Proceedings of Computer Assurance (COMPASS), pp. 109–122 (1995)
8. Heitmeyer, C., Bharadwaj, R.: Applying the SCR Requirements Method to the Light Control Case Study. Journal of Universal Computer Science 6(7), 650–678 (2000)
9. Jackson, M.: Problem Frame: Analyzing and Structuring Software Development Problems. Addison-Wesley (2001)
10. Kang, K.C., Kim, S., Lee, J., Shin, E., Huh, M.: FORM: A Feature-oriented Reuse Method with Domain-specific Reference Architecture. Annals of Software Engineering 5, 143–168 (1998)
11. Kang, K.C., Lee, J., Donohoe, P.: Feature-Oriented Product Line Engineering. IEEE Software 9(4), 58–65 (2002)
12. Lee, K., Kang, K.C.: Usage Context as Key Driver for Feature Selection. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 32–46. Springer, Heidelberg (2010)
13. Leveson, N.G.: Safeware: System Safety and Computers. Addison-Wesley Publishing Company (1995)
14. McCarthy, J., Hayes, P.J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence 4, 463–502 (1969)
15. Parnas, D.L., Madey, J.: Functional Documentation for Computer Systems Engineering, McMaster University, Technical Report CRL 237 (1991)

16. Reiser, M.-O., Weber, M.: Using Product Sets to Define Complex Product Decisions. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 21–32. Springer, Heidelberg (2005)
17. Tun, T.T., Boucher, Q., Classen, A., Hubaux, A., Heymans, P.: Relating Requirements and Features Configurations: A Systematic Approach. In: Proceedings of the 13th International Software Product Line Conference (SPLC 2009), pp. 201–210 (2009)
18. Ubayashi, N., Otsubo, G., Noda, K., Yoshida, J.: An Extensible Aspect-Oriented Modeling Environment. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 17–31. Springer, Heidelberg (2009)
19. Ubayashi, N., Nakajima, S., Hirayama, M.: Context-Dependent Product Line Practice for Constructing Reliable Embedded Systems. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 1–15. Springer, Heidelberg (2010)