International Journal of Hybrid Information Technology
Vol.8, No.3 (2015), pp.21-32
http://dx.doi.org/10.14257/ijhit.2015.8.3.03

Adaptation Model Construction Technique Based on Partial Behaviour Model Fusion

Xiaoan Bao, Hui Lin, Ning Gui, Xiance Sun, Na Zhang and Meng Dong

Zhejiang Sci-Tech University, Hangzhou 310018, China
baoxiaoan@zstu.edu.cn

Abstract

Nowadays, traditional model construction techniques are often used in the self-adaptive software development environment. However, due to its adaptation problems, such as low reusability and high complexity, it is difficult to meet the incremental requirements of users. In order to achieve the combination of models and the results verified, this paper regards improving the reusability of adaptation models as a starting point, introducing the formal method of partial behaviour model to the description of adaptation behaviour. With the help of three-valued logic model description language KMTS, we research the consistent judgment algorithm of related models, and propose a fusion method of consistent models for supporting online fusion of adaptation models. Finally we use a model instance and its application to analyze and verify the correctness and effectiveness of the fusion result.

Keywords: *model fusion, adaptation model, KMTS, consistent model*

1. Introduction

In the current self-adaptive software design process, it mainly employs dedicated adaptation module construction technique in order to meet the increments of software development mode. This mode requires multiple modules together to achieve users' requirements, when the users' requirements or environment changes, module replacement even regroup will be used to complete the overall design. However, due to the different design goals or description languages, the combination design of adaptation logic between the existing software module and the requirement module is very difficult. In the self-adaptive software design, adaptation logic is regarded as the brain of self-adaptive software evolution, which plays an important role when software system adapts to the changeable environment. The adaptation logic of self-adaptive software usually has a strong environmental coupling. This means when the software operating environment is inconsistent with its original environment, the adaptation mechanisms will be adjusted accordingly to adapt to new environment. Oreizy, *et al.*, [1] have stated: "In order to adapt to changing environmental requirements, not only adaptation software should be changed to reflect the new environmental requirements, the system adaptation logic also must evolve accordingly."

Some well-known researchers have in-depth research in the adaptation logic combination, such as Maurel, *et al.,* [2] proposed a speculation-based integration strategy, the core of which is the assumption that each adaptation logic will not conflict, so each adaptation project can be directly superimposed under the assumption. Sicard, *et al.*, [3] built a self-adaptive software framework supporting software self-healing, in which the fusion method of two different adaptation goals is to design a special algorithm in order to eliminate possible conflicts. However, this dedicated algorithm constitutes a strong coupling mechanism in the middle of two different adaptation logics, making the adaption module difficult to be reused in different contexts. Jie, *et al.,* [4] realized a self-adaptive mechanism based on Autonomous components of run-time quality evaluated dynamically, through monitoring the performance of Autonomous components to evaluate the merits of self-adaptive strategies and make adjustments, but it did not address how to build the process of adaptation logic. These self-adaptive software development patterns based on multi-model fusion are usually solved with empirical methods, so it is difficult to verify the correctness and validity of the merged modules.

In order to solve verification issues after module combination better, with the help of partial behaviour model [5] mathematical tool, the formal method of partial behaviour model is introduced into the adaptation description of self-adaptive software to make the process of adaptation logic combination formalized. By accurately describing the unknown and the re-configurable information of modeling, it makes the model fusion and its verification possible. It regards different self-adaptive software modules as different adaptation models, through the continuously fusion of partial behaviour models to meet the incremental requirements. On the basis of the formal description, we design the consistent judgment algorithm between partial adaptation models and adaptation logic fusion method to provide reliable support for the fusion of the consistent models. Through building the appropriate model fusion instance, abstract model analysis and the actual development application, we realize the accuracy validation of fusion results.

2. Model Fusion Design

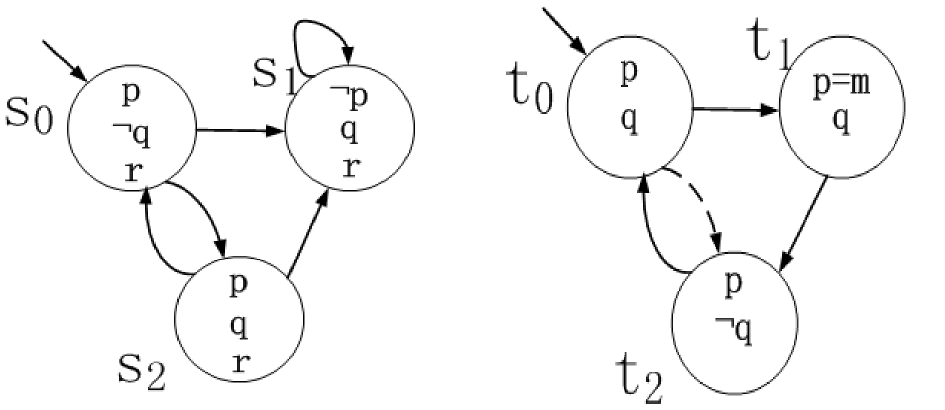
2.1. Analysis of Adaptation Behavior Formal Description

In order to describe the process of self-adaptive software evolution accurately, the researchers have proposed a lot of formal description methods, and used the relevant mathematical tools to analyze the characteristics of evolution. For example, Oreizy, *et al.*, [6] employed the C2 ADL language, and made use of Archstudio development and runtime environment to support dynamic modification of software; Li, *et al.*, [7] proposed a theoretical model of dynamic self-adaptive systems and defined self-adaptive software architecture with formal methods. However, the formal descriptions they have studied essentially focus on the changes of the architecture. Zhang, *et al.*, [8] pointed out that self-adaptive software design usually involves the changing of the system adaptation behavior, and used A-LTL to describe adaptation behavior based on temporal logic approach. Kramer, *et al.*, [9] used π calculus to represent the dynamic behavior of the software in order to describe the behavior evolution of self-adaptive software; Li, *et al.*, [10] proposed a dynamic architecture description language D-ADL, and formalized the dynamic behavior to high-order π calculus derivation, which can be used to formally describe the dynamic change process of system architecture in design pattern. These studies usually just change the system dynamical adaptation behavior into a complete adaptation model, but the model is not well to support the online construction mechanism based on the partial adaptation model combination.

Some researchers have proposed the concept of partial behavior model to support incremental software development patterns. It well supports the separation of concerns development pattern, and gradually becomes a research hot of behavior model formalization. Nowadays, there are many formal languages and architectures based on partial behavior model, such as Partial Labelled Transition Systems [11], Multi-Valued State Machines [12] and so on. These studies support the modeling of partial behavior model and the description of unknown information, and some non-deterministic description can be stepwise refinement in the design of behavior models or gradually removed in multi-part model fusion process, thus it provides a well support for the fusion description of adaptation behavior models.

2.2. Selection and Analysis of Formal Description Language

In order to realize the incremental fusion of adaptation models and resolve conflicts arising from the fusion process, adaptation models must support explicitly modeling for the current unknown information. The traditional modeling language kripke structure, as shown in Figure 1(a), uses Kripke structure to define all the states and transitions, but does not support the possible state tr'[\ansitions or refinement. While Figure 1(b) shows three-valued logic Kripke Modal Transition System(KMTS), supporting the may transitions and the possible states (p=m means the p event is unknown), so it can describe the unknown and reconfigurable information effectively.



a. A Kripke Structure b. A Kripke Modal Transition System (KMTS)

Figure 1. Examples of State-based Formalisms

A KMTS is a tuple $(S,s_0,R^{must},R^{may},L,AP)$, and defines all states and transitions, including may states and may transitions, where S is a set of states,s_0 is the initial state, $R^{must} \subseteq S\times S$ and $R^{may} \subseteq S\times S$ are must and may transition relations, respectively. $L\rightarrow 3^{AP}$ is a three-valued labelling function; using elements true (t), false (f) and maybe (m) to express whether the event is happening. AP represents the set of events in a model, while in a merged model, AP_u is used to contain all events.

Figure 1(b) shows an example of a KMTS, where

1) $S=\{ t_0 ,t_1 ,t_2\}$,and t_0 is the initial state;

2) $R^{must}=\{ (t_0 ,t_1),(t_1 ,t_2),(t_2 ,t_0)\}$;

3) $R^{may}=\{ (t_0 ,t_1),(t_1 ,t_2),(t_2 ,t_0), (t_0 ,t_2)\}$;

4) $L(t_0)=\{(p,t),(q,t)\}$, $L(t_1)=\{(p,m),(q,t)\}$, $L(t_2)=\{(p,t),(q,f)\}$;

5) $AP=\{p,q\}$

In a KMTS description, we define $R^{must}\subseteq R^{may}$, meanwhile, transitions represented by solid arrows are in R^{must}, and transitions represented by dotted arrows are in $R^{may}\setminus R^{must}$.

2.3. Model Fusion Process

In the self-adaptive software development, users' requirements and self-adaptive environment are often subject to changes, and the relationship between the models is difficult to determine in advance, which affects the fusion strategies and fusion effects of models. Model fusion process studied in this paper is aimed at consistent models, meanwhile, the environment and requirements should keep unchangeable to ensure stability.

The fusion of two adaptation models, shown in Figure 2, needs a series of the fusion steps. Firstly, we need to determine the relationship between the models. Due to the different design objectives and design points of view, different models may use different description languages, so it is difficult to determine the relationship between each other. This paper describes a unified formal language KMTS for all models, and synthetic judges whether the models are related according to the operating mode of state machine. Among related models, they can be divided into inconsistency and consistency based on the presence or absence of conflicts between each other. Secondly, for the relationship between the models, we should take different fusion strategies [13] to compare and merge these related models. When the consistent models are fusing, the key argument is to determine the states and the transfer relationships with the same property, providing the starting point for the fusion behavior. Finally, we need to analyze the results of the fusion to ensure that the model has achieved the fusion targets.

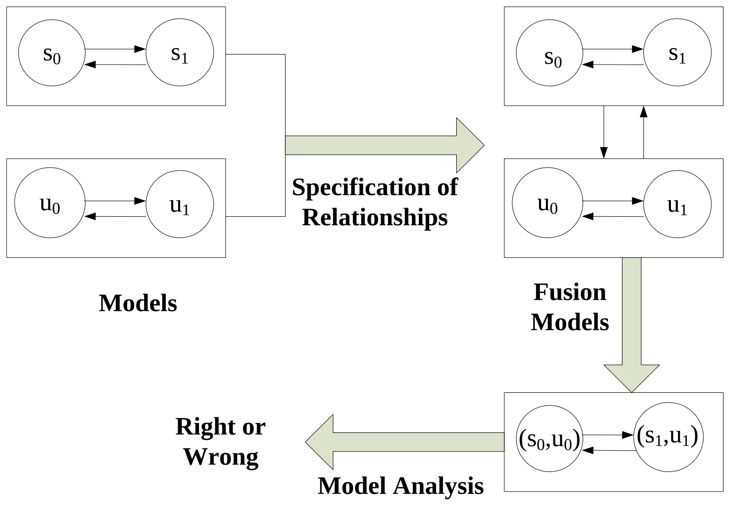


Figure 2. Overview of the Model Fusion

3. Adaptation Model Fusion Algorithm

The process of model fusion is to compute the least common refinement (LCR) [14] between models. Prior to the model fusion, we need to describe each model with KMTS. If KMTS description languages between models are independent, it can be considered there is no interference during model fusion, and the LCR generated is unique, which can be regarded as a result of model fusion. Otherwise we may have several incomparable minimal common refinements described in KMTS. Thus this paper has a further research to obtain the least common refinement.

3.1. Unified Description

Adaptation models waiting for fusion are all inherently incomplete, and each model just focuses on a few features of a system. For example, we suppose there are two models K_1 and K_2, thus KMTS formal description is shown as follows:

K_1:{S_1={ s_0 ,s_1 }; R^{must}={ (s_0 ,s_1) }; R^{may}={ (s_0 ,s_1) }; L(s_0)={(p,t)}, L(s_1)={(p,t); AP={p}}

K_2: {S_2={ u_0 ,u_1 ,u_2}; R^{must}={ (u_0 ,u_1) ,(u_1 ,u_2)}; R^{may}={(u_0 ,u_1) ,(u_1 ,u_2)}; L(u_0)=(q,f), L(u_1)=(q,f), L(u_2)=(q,f); AP={q}}

We denote the set {p, q}, the unified set of events for the system, by AP_u. In order to compare and merge these incomplete models, we need to unify their descriptions. The unification of model descriptions can be divided into two parts as follows:

1) Unified behavior. In a single KMTS model description, all state transitions are certain, so R^{must} is equivalent to R^{may}. Each model only describes the states relevant. However, in a unified description, each model must describe all possible states. Hence we need to increase the possible states and may transitions which have not been described. In a model K_1,in order to keep corresponding with the states and transitions of K_2,we modify the KMTS partial descriptions are $\{S_1=\{ s_0 ,s_1 ,s_2\}; R^{must}=\{ (s_0 ,s_1) \}; R^{may}=\{ (s_0 ,s_1),(s_1 ,s_2)\}\}$.

2) Unified events. For each state in the model, its inner events should be compared with APu and made adjustments in order to keep same with AP_u. For the complementary events, their properties values are marked m, which means that the events may happen. After adding q and p to K1 and K2 respectively, the modified KMTS descriptions are shown as follows:

K_1: $L(s_0)=\{(p,t), (q,m)\}$, $L(s_1)=\{(p,t), (q,m)\}$,$L(s_2)=\{(p,m), (q,m)\}$; $AP=\{p,q\}$

K_2: $L(u_0)=\{(p,m), (q,f)\}$, $L(u_1)=\{(p,m), (q,f)\}$,$L(u_2)=\{(p,m), (q,f)\}$; $AP=\{p,q\}$

3.2. The Judgment of Consistent Models

Suppose there are two models K_1 and K_2. Unify their KMTS descriptions, then judge whether K_1 and K_2 are consistent. We define a consistency tag $\sim \subseteq S1\times S2$, while s and u represent the states from K_1 and K_2 respectively, if $s \sim u$, they should satisfy the following conditions:

1)$\forall p\in AP_u \cdot (L_1(s,p)=t \Rightarrow L_2(u,p)\neq f)\wedge (L_2(u,p)=t\Rightarrow L_1(s,p)\neq f)$

2) $\forall s'\in S_1 \cdot R_1^{must}(s,s')\Rightarrow \exists u'\in S_2 \cdot R_2^{may}(u,u')\wedge s'\sim u'$

3) $\forall u'\in S_2 \cdot R_2^{must}(u,u')\Rightarrow \exists s'\in S_1 \cdot R_1^{may}(s,s')\wedge s'\sim u'$

The mark p represents the event from AP_u ,while s′ and u′ represent the states of K_1 and K_2 respectively. We can infer that the consistent model judgment above is recursive. If $s\sim u$, then all events in these states are consistent, and they have consistent successors. Thus, if initial states s_0 and u_0 satisfy $s_0\sim u_0$,we can infer $K_1\sim K_2$, means the two models keep consistent. We have used a greatest fixpoint to find a consistency relation between two models, and the algorithm is shown in Figure 3.

```
Algorithm. Consistency Relation Judgment
Input: Partial behaviour models K1 and K2 with state spaces $\Sigma_1$ and $\Sigma_2$.
Output: A consistency relation  $r \subseteq \Sigma_1 \times \Sigma_2$.
1:  $r=\Sigma_1 \times \Sigma_2$
2: do
3:   changed = false
4:   for every $(s, t) \in r$ :
5:       if s and t disagree on the value of some event $p \in AP_u$ :
6:           $r = r \setminus \{(s, t)\}$ // s and t are inconsistent
7:           changed = true
8:       if  $\exists s' \cdot R_1^{must}(s,s') \wedge \forall t' \cdot R_2^{may}(t,t') \Rightarrow (s',t') \notin r$:
9:           $r= r \setminus \{(s, t)\}$ // s and t are inconsistent
10:          changed = true
11:      if $\exists t' \cdot R_2^{must}(t,t') \wedge \forall s' \cdot R_1^{may}(s,s') \Rightarrow (s',t') \notin r$:
12:          $r= r \setminus \{(s, t)\}$ // s and t are inconsistent
13:          changed = true
14: while changed :
15: if $(s_0, t_0) \in r$ :
16:   return r // consistency relation is computed
17: else
18:   return  $\Sigma_1 \times \Sigma_2 \setminus r$ // K1 and K2 are inconsistent
```

Figure 3. Algorithm for Computing a Consistency Relation

3.3. Consistent Model Fusion Method

After unifying the KMTS description of models and making sure the models are consistent, we define a merge of K_1 and K_2 , denoted K_1+K_2, as a tuple $(S_+,(s_0,u_0),R_+^{must},R_+^{may},L_+,AP_1 \cup AP_2)$,which satisfies the following conditions:

1) $S_+ = \{(s,t)| s \sim u\}$

2) $R_+^{must}=\{((s,u),(s',u'))|(R_1^{must}(s,s') \wedge R_2^{may}(u,u')) \vee (R_1^{may}(s,s') \wedge R_2^{must}(u,u'))\}$

3) $R_+^{may}=\{((s,u),(s',u'))|(R_1^{may}(s,s') \wedge R_2^{may}(u,u'))\}$

4) $\forall p \in AP_u \cdot L_+((s,u),p)=L_1(s,p) \nabla L_2(u,p)$

Symbol ∇ represents the union set of information volume, while the information volume of property value t and f contained herein both are greater than m, then $t \nabla m=t$ and $f \nabla m=f$. When the two consistent states are merging, each event should be considered, and given the property values after the fusion. For the transitions relations between states, we need to distinguish the must and may transitions.

In summary, the proposed algorithm flowchart of consistent models based on the partial behaviour models is shown in Figure 4.

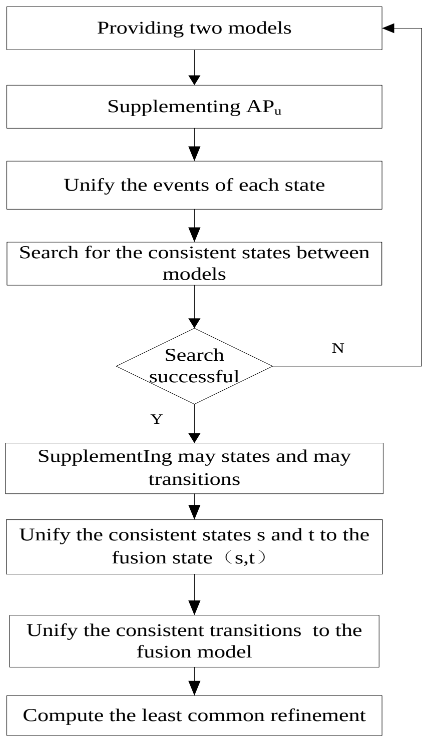

Providing two models
Supplementing AP$_u$
Unify the events of each state
Search for the consistent states between models
Search successful
N
Y
SupplementIng may states and may transitions
Unify the consistent states s and t to the fusion state（s,t）
Unify the consistent transitions to the fusion model
Compute the least common refinement

Figure 4. The Flowchart of Behavior Model Fusion

In the fusion process, the key steps are following:

1) The original model and requirement model have a breadth-first search, and unify the set of events in the states, so that each state contains all events keeping same with AP$_u$. For the added event p, we use m to represent its property value.

2) Regard the initial state s_0 of original model as a starting point, traversing all the states of requirement model to find a consistent state, and vice verse. If none of the successful search, then the two models is inconsistent, fusion method cannot be used here. If success, then add the corresponding states between the two models, and the dashed arrows indicate the may transitions.

3) Unify the events of consistent states s and t to the merged state (s,t), and unify the consistent relations to the merged model.

4) Compute the least common refinement of the merged model, and analyze the events with the value m, then modify their values to t or f. After model fusion is over, verify the corresponding adaptation logics.

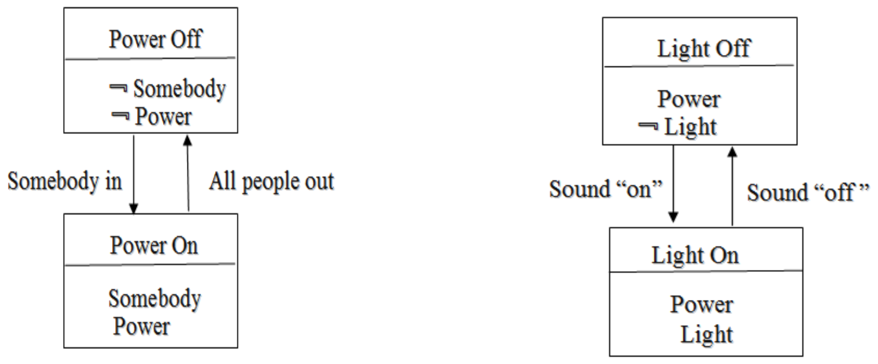
4. Case Study and Analysis

4.1. Case Study

In a typical adaptation module combination system, for example, the entire control system contains "Data Acquisition" and "Power Control" two modules. Power switch is in control of a sensor. When someone enters the room, the power is automatically turned on, otherwise off. To solve the lighting problem in the room, the user increases the requirement to control the light switch through voice commands. Therefore, the control strategy needs to adjust to suit the requirement of jointing control of power and lighting.

There are two events in the power control model, namely the power event and somebody event. The model uses **Power** and **Somebody** to represent whether the power source is on and whether there is anyone in the room respectively. The

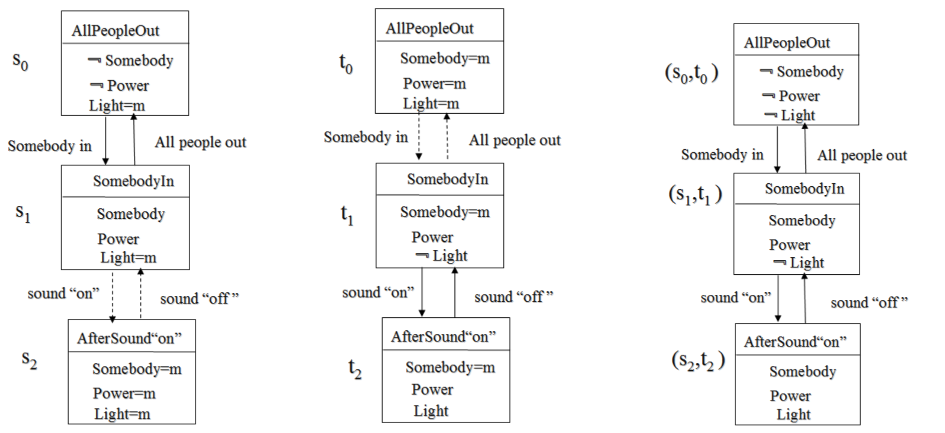
KMTS description is shown in Figure 5 (a). For the voice-activated requirement model, in a similar way, it uses **Light** to represent whether the lights are turned on. When the power is turned on in advance, sound "on" can make the lights on, and the sound "off" then makes the lights opposite. The KMTS description of the requirement model is shown in Figure 5 (b).



a. Power Control Model b. Voice-activated Requirement Model

Figure 5. Single KMTS Description

However, in all states of Figure 5(a), they include only the events **Somebody** and **Power**, while lack the event **Light** and may transitions response to the sound. Therefore, the set of vocabulary of power control model should be extended to AP_u by setting the missing event **Light** to m in all states of the model, as shown in Figure 6(a). Similarly, Figure 6(b) is a unified KMTS description of voice-activated requirement model.



a. Power Control Model b. Voice-activated Requirement Model c. Model Fusion Result

Figure 6. Unified KMTS Description

After unifying the KMTS description of the power control model and voice-activated lighting requirement model, according to adaptation model fusion algorithm, we can determine the consistency relationship between the two models, and conduct model fusion. The fusion result is shown in Figure 6(c).

4.2. Analysis of Fusion Results

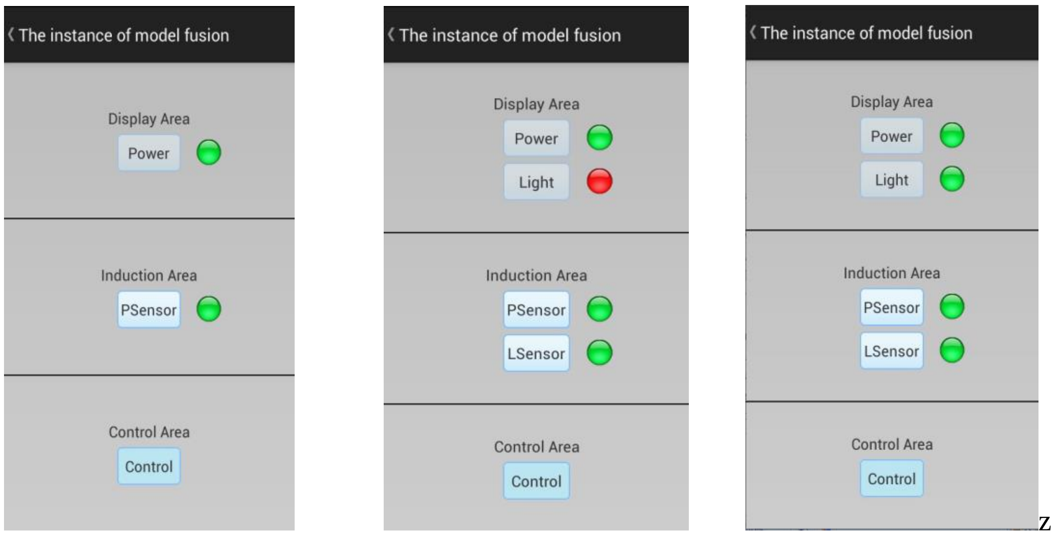
In the premise of understanding the results of model fusion, we can analyze it from constraints and scenes two aspects.

In a model, each event can be seen as its behaviour, and the events in the merged model must be associated. The fusion purpose of power and lighting models is to meet user's requirement for room lighting. While for the each state of merged model, the number of events has a minimum and maximum constraint. Generally, few states can include all events. Thus, for all the states valued m in the merged model, if they cannot be captured the exact value from the connected events, we will have to remove them from the states. In this example, after the model fusion, the value of **Light** event in the **All People Out** is m, but it can be analyzed to obtain its exact true value is f, therefore it should be retained. Meanwhile, for the same event, it shouldn't be repeated in the same state.

The original model and requirement model represent different scenes. Such as power control model is to meet the power needs in the room, and the voice-activated lighting requirement model is to meet the lighting need in the room. It can be found from the scene analysis that the power source should be turned on in advance; otherwise the light control cannot be available. Therefore, when the model fusion is over, fusion results must conformed to their relations.

4.3. Development Show of Model Fusion

We use android platform development tool to simply develop the case in this paper, and the interface is shown in Figure 7.The whole system is divided into three main parts: 1) Display area. You can display the power and lighting event with indicator light showing their status. In real life, only the display area is presented in front of people. 2) Induction area. It consists of various types of sensors with indicator lights to show the sensor status. For example, when the power sensor turns green, it means someone enters the room; similarly, light sensor can sense the incoming sound. 3) Control area. Based on the feedback from the sensing area, it uses fixed adaptation logic to control the corresponding events. Control area is in the main centre of the entire system, and plays a vital role in the operation of the system.



a. The Power Control Along   b. The Voice-activated Light Can't Work   c. The Perfect Control

Figure 7. The Instance of Model Fusion

Figure 7(a) shows that the whole system only need to implement the power control function, and the inherent adaptation logic in control area is to control the power status according to the status of power sensor. In the case of constant adaptation logic, the power is consistent with its corresponding sensor. After adding voice-activated lighting requirement, as is shown in Figure 7(b), if not carry out model fusion or the fusion result is incorrect; the adaptation logic in control area is also incorrect. Therefore, even if the light sensor senses the incoming command sound, the corresponding light cannot be turned on, and the colour of light next to it is still red. Only adaptation logic changes right can make the control function of the system run successfully, as shown in Figure 7(c).

In actual operation, the power switch and voice-activated lighting are abstracted into two behaviour models for modeling. Through the fusion algorithm, if they are able to fuse with each other, their adaptation logic will also fuse simultaneity. The adaptation logic of complete merged model will be able to correctly and effectively control the entire system.

5. Conclusion

In the self-adaptive software development process, module combination can effectively solve complex modeling problems. However, due to the adaptation logic of dedicated modules accompany with high complexity, low reusability and other issues, it is difficult to verify the effectiveness and correctness of the combination result. In order to solve the problems, this paper introduces the formal method of partial behaviour model to the descriptions of adaptation behaviour to analyze the dynamic combination process of adaptation logic based on system local behaviour, making the model fusion formalized and provides a strong foundation for the validation of fusion results. KMTS description language used in this paper supports may behaviours and may states and can effectively provide the unknown and reconfigurable description information. Thus, when the two KMTS models are consistent, their fusion can be regarded as the process of computing the least common refinement. The consistent model judgment and fusion algorithm in the paper also provide an effective basis for the validation of the merged model.

The model fusion method studied in this paper is mainly for the consistent adaptation models. Meanwhile, the model fusion based on fixed requirements need to maintain a stable self-adaptive environment. We will carry out further research on the fusion between inconsistent models and adaptation conditions of the changing environment.

Acknowledgements

This work was supported by National Natural Science Foundation of China (61202050/61379036), Zhejiang Provincial Natural Science Foundation of China (Y13F020175), Zhejiang Talent Project (2013R10005), the New-shoot Talents Program of Zhejiang Province (2014R406073), Zhejiang High-skilled Talent Project (2013R30019) and 521 talent project of ZSTU.

References

[1] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum and A. L. Wolf, "An architecture-based approach to self-adaptive software", IEEE Intelligent Systems and their Applications, vol. 14, no. 3, **(1999),** pp. 54-62.

[2] Y. Maurel, A. Diaconescu and P. Lalanda, "Creating Complex, Adaptable Management Strategies via the Opportunistic Integration of Decentralised Management Resources", Proceedings of the 2009 International Conference on Adaptive and Intelligent Systems, **(2009)** September 24-26, Klagenfurt, Austria.

[3] S. Sicard, F. Boyer and N. De Palma, "Using components for architecture-based management: the self-repair case", Proceedings of the 30th International Conference on Software Engineering, **(2008)** May 10-18, Leipzig, Germany.

[4] J. J. Jie, T. X. Shi, W. P. Jiao and F. J. Meng, "Autonomous components whose adaptation policies can be customized online and evaluated dynamically", Journal of Software, vol. 23, no. 4, **(2012),** pp. 802-815.

[5] S. Uchitel, D. Alrajeh, S. Ben-David, V. Braberman, M. Chechik, G. D. Caso, N. Dlppolito, D. Fischbein, D. Garbervetsky, J. Kramer, A. Russo and G. Sibay, "Supporting incremental behaviour model elaboration", Computer Science-Research and Development, vol. 28, no. 4, **(2013),** pp. 279-293.

[6] P. Oreizy, N. Medvidovic and R. N. Taylor, "Runtime Software Adaptation: Framework, Approaches, and Styles", Companion of the 30th international conference on Software engineering, **(2008)** May 10-18, Leipzig, Germany.

[7] Y. Li and Z. H. Wu, "Research on architecture of dynamic self-healing system", Journal of Zhejiang University (Engineering Science), vol. 39, no. 2, **(2005),** pp. 216 - 220.

[8] J. Zhang, B. H. C. Cheng, "Using temporal logic to specify adaptive program semantics", Journal of Systems and Software, vol. 79, no. 10, **(2006),** pp. 1361 - 1369.

[9] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge", Future of Software Engineering, **(2007)** May 23-25, Minneapolis, MN, USA.

[10] C. Y. Li, G.S. Li and P. J. He, "A formal dynamic architecture description language", Journal of Software, vol. 17, no. 6, **(2006),** pp. 1349 - 1359.

[11] S. Uchitel, J. Kramer and J. Magee, "Behaviour model elaboration using partial labelled transition systems", ACM SIGSOFT Software Engineering Notes, vol. 28, no. 5, **(2003),** pp. 19 - 27.

[12] R. P. D. Redondo and J. J. P. Arias, "Reuse of verification efforts and incomplete specifications in a formalized, iterative and incremental software process", Proceedings of the 23rd International Conference on Software Engineering, **(2001)** May 12-19, Toronto, Canada.

[13] L. Yi, H. Y. Zhao, W. Zhang, Z. Jin and H. Mei, "Research on the merging of feature models", Chinese Journal of Computers, vol. 36 no. 1, **(2013),** pp. 1 - 9.

[14] D. Fischbein and S. Uchitel, "On correct and complete strong merging of partial behaviour models", Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, **(2008)** November 9-14, Atlanta, GA, USA.