Encoding Scenarios with Design Information Framework for the Generation of Multiple Aspect Models for System Implementation

Youn-kyung Lim
Institute of Design, Illinois Institute of Technology
350 N. LaSalle St.
Chicago, IL 60610 USA

and

Keiichi Sato
Institute of Design, Illinois Institute of Technology
350 N. LaSalle St.
Chicago, IL 60610 USA

ABSTRACT

Multidisciplinary work is inevitable in designing a complex system. A system design is the output of integrating multiple aspects from the multidisciplinary teamwork. However, the communication and association between those multiple aspects has been one of the most difficult tasks in design. By introducing the Design Information Framework (DIF) as a unified information platform, sharing and interpreting multiple aspects from different disciplines was enabled. Scenarios have been used for describing new situations with new system concepts. DIF provided a mechanism of deconstructing scenarios to generate multiple aspects that could effectively be associated with each other. A case study of designing a cellular phone in the context of its use in a car environment was conducted to demonstrate this mechanism and function of DIF for generating multiple aspects for system implementation.

Keywords: Scenario-Based Design, Information Systems, System Design, System Implementation, Multidisciplinary Teamwork

1. INTRODUCTION

The system development process requires collaboration among multidisciplinary team members in natural sciences, business, management, marketing, art, and others. The multidisciplinary teamwork is even more significant when design problems become complex [7]. In such situations, the integration of multidisciplinary information is required to satisfy diverse requirements of developing systems. As systems become more complex, a large amount of information is necessary for the development, and one system may consist of multiple products. In this research, a mechanism of representing information from multidisciplinary teams for effectively interpreting each other's concept in the system development process has been developed. Design Information Framework (DIF) was introduced as a basic

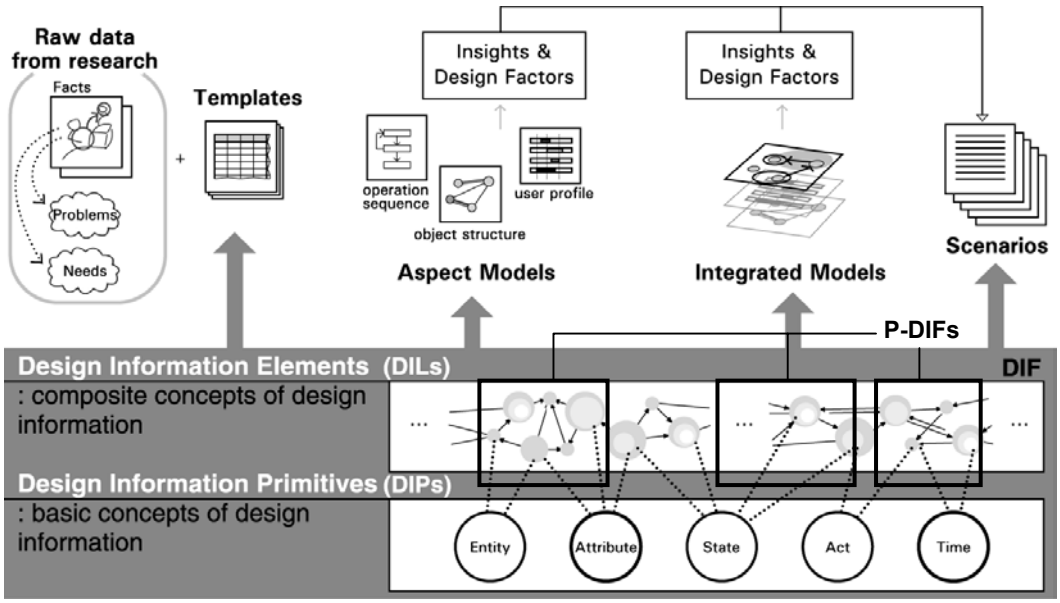

Figure 1: DIF in a design process as a platform

platform for this mechanism. It is a unified and shared information platform to enable system developers to organize and manipulate information throughout the development process (Figure 1). It can then be used to generate models that represent various aspects of information created by multidisciplinary teams. We decided to call the model that represents each aspect as "aspect model."

DIF represents information on two levels. The lower level of representation consists of Design Information Primitives (DIPs) that cannot be further decomposed into smaller conceptual units. The higher level of representation consists of Design Information Elements (DILs) which are composed of those DIPs [6]. The information elements in DIF therefore are DILs and DIPs (Figure 1).

DIF has an open structure that does not limit the possible types of information. Each design project requires a different set of information elements to be represented in DIF, and this set is called as Project-based DIF (P-DIF). A P-DIF can be determined by the nature of the project and the applied methodology. For example, if a scenario-based design method is used for an interactive system design project, a P-DIF for that project can include the information elements such as *goals*, *activities*, *settings*, and *conditions*, which correspond to theatrical play scripts that are structured with actors, events, scenes, and props.

In the domain of software development, multi-disciplinary teamwork is essential. Scenarios are one of effective means for communication because of their format understood by every audience. There have been many attempts to deconstruct scenario content for the use of the system implementation purpose by narrowing the gap between specification and implementation. This requires integration of various aspects of system development [3]. However, the deconstructed contents are used only for specific purposes or aspects of system implementation. This makes it very difficult to associate all dispersed methods for system implementation.

In this regard, the scenario encoding mechanism utilizing DIF can solve this problem because it provides a generic format for encoding and re-representing the narrative data in various formats. This is important in terms of diverse teams' participation in the implementation process of a system, such as human factor, AI, system engineering, and software engineering teams. Their ability to communicate is one of the key issues for the integration of their outputs for the system development.

A case study was conducted to develop and demonstrate the mechanism for this purpose. For this case, cellular phone use in a vehicle is selected to analyze the cognitive load of the driver. It requires various aspects of a system for its implementation such as product-hardware and product-software. In this case, how DIF supports the construction of consistency between various aspects of product implementation through scenarios is the key issue. This case study demonstrates how the DIF-enabled mechanism of aspect model construction can easily transform one aspect model to another in order to facilitate effective communication between development team members who may have different disciplinary backgrounds.

2. SCENARIO CONSTRUCTION MECHANISM WITH MULTIPLE ASPECT MODELS

Before introducing the case study, the mechanism of constructing scenarios with a P-DIF and aspect models should be explained. P-DIFs provide parameters to construct appropriate aspect models, and the aspect models then inform the content of a scenario when re-organized into building blocks (Figure 2). A typical scenario building block is defined by one of the following four information categories [2][3]:

- **Actor Profile**: description about agents or actors who are the main subjects of the scenarios
- **Scene Setting**: description of the relationships among physical elements in the situation or among the actors involved in the scenario
- **Goals**: what the actors want to achieve
- **Events**: the main contents of the scenarios, describing actors' actions

For example, a layout model describes environmental information in a scenario. User profiles and a role-activity model provide information about actors such as their jobs, roles, and basic relationships with other actors. This information sets the scene for the scenario and provides background information on actors. When the scene influences the outcome of events, it is also embedded into the main story.

The body of the scenario is constructed using events and actors' goals. Goals drive actors' basic behaviors in a given event, and every scenario involves at least one actor and one goal. A hierarchical task analysis (HTA) model provides the structure of the event in the scenario along with an actor's goal and task structure.

In addition to using aspect models as references for basic contents of a scenario, another benefit of using these models lies in generating further insights from the integration of the models. The problems found in a use situation can be embedded in part of a scenario. The problem identified in a specific aspect, such as a flow of information, can be viewed in other aspects, like an activity sequence or a spatial setting. Through this integration process, we can systematically interpret a problem in a holistic way.

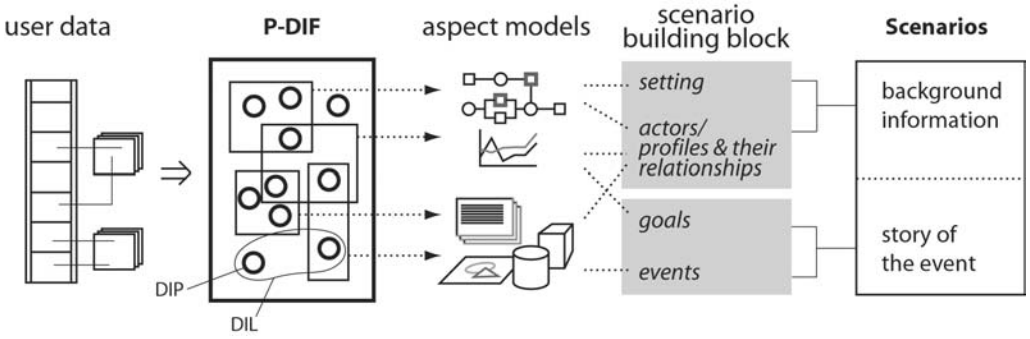

Figure 2: The Process of Generating Scenarios with DIF

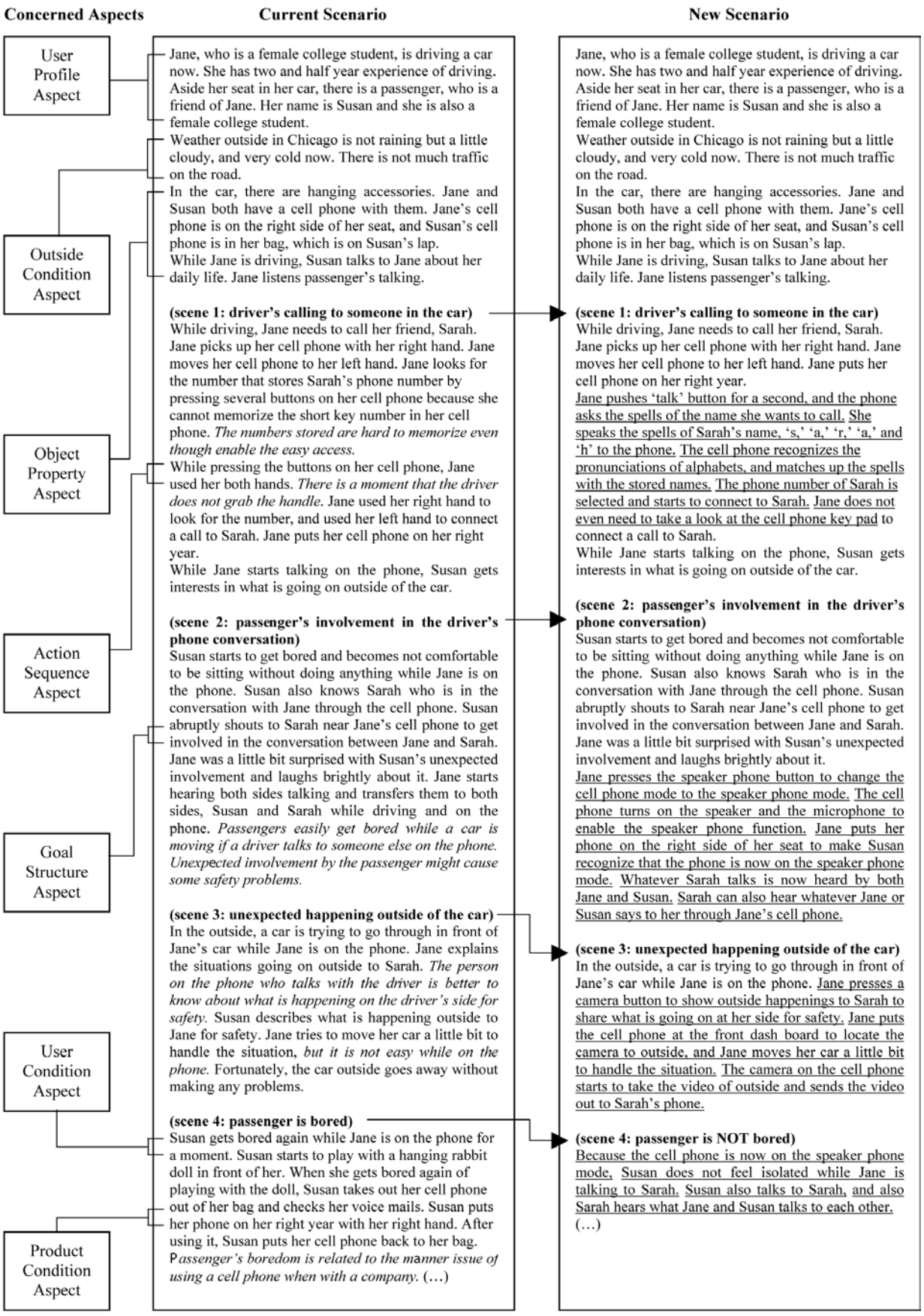

Figure 3: Current Scenario and New Scenario of Using a Cellular Phone in a Car

Table 1: A P-DIF for Encoding the Scenario to Construct the Selected Models

Selected models	DILs (DIPs or DILs)	A part of encoded data from the scenario
Task Knowledge Structure (TKS)	Object Description (Entity, Attributes, Genericity Attribute, Centrality Attribute)	Obj_1 (address book, easy-to-access, specific, crucial) Obj_2 (speaker, small, specific, optional) Obj_3 (microphone, small, specific, optional) …
	Goals (Agent Entity, Act, Object Entity, Time)	Goa_1 (driver, connect, a call to someone, while driving) Goa_2 (driver, pull out, phone number, while driving) …
	Goal Relationships (Goal, Sub-goal)	GoR_1 (Goa_1, Goa_2) GoR_2 (Goa_2, Goa_3) GoR_3 (Goa_2, Goa_6) …
	Actions (User Entity, Act, Object Entity, Tool Entity, Space Entity, Time)	Acn_1 (driver, push, 'talk' button, thumb of right hand, for a second) Acn_2 (driver, speak, the spells of call-receiver's name, the phone) …
Classes-Responsibilities-Collaborations (CRC) model	Collaborations (Class Entity, Collaborating Entity)	Col_1 (alphabet sound recognizer, voice receiver) Col_2 (alphabet interpreter, alphabet sound recognizer) …
	Responsibilities (Class Entity, Function, Collaborating Entity)	Res_1 (alphabet sound recognizer, recognize alphabet pronunciations, microphone) Res_2 (alphabet interpreter, translate alphabet sounds to alphabet letters, alphabet sound recognizer) …
Functions-Behaviors-States (FBS) model	Functions (Product Entity, Act, Object Entity, Act Attribute)	Fun_1 (cell phone, ask, the spells of the name to call) Fun_2 (cell phone, recognize, the pronunciations of alphabets, clearly …
	Function Relationships (Function, Relationship Attribute, Related Function)	FuR_1 (Fun_3, conditioned-by, Fun_2) FuR_2 (Fun_2, conditioned-by, Fun_1) FuR_3 (Fun_4, decomposed-into, Fun_5) FuR_4 (Fun_4, decomposed-into, Fun_6) …
	Function-Object Relationships (Function, Entities)	FuO_1 (Fun_2, alphabet sound recognizer) FuO_2 (Fun_3, alphabet interpreter) FuO_3 (Fun_5, speaker) …

3. COMPARABLE SCENARIOS FOR THE CURRENT SYSTEM AND THE NEW SOLUTION

For the case study, a current situation scenario was created based on the scenario development mechanism explained in the previous section. A scenario should include the background information and the story of an event. The background information was constructed according to the condition/status aspect of three domains: a driver, a vehicle, and driving environment. Driver variables included level of training, experience, attentiveness, or driving time. The event part of the scenario was constructed on the structure of tasks related to cellular phone use. Figure 3 shows both current situation scenarios and new situation scenarios focusing on the use of a cellular phone while driving. General problematic conditions of driving while using a cellular phone such as driving in unfamiliar territory, driving in fast moving traffic, or driving while conversing with a passenger, were applied as a basic context of each scenario. Requirements for designing a cellular phone in the vehicle environment for those situations were also described in the scenarios.

When creating a scenario about a new system, it is a good idea to create it parallel to the scenario of the current system so that the two situations can be compared effectively as well as providing a guideline for developing the new scenario. Because the current scenario was created by considering multiple aspects of the situation and represents important problems and requirements, the scenario for the new system can contain a variety of aspects of the situation addressed by the current one. The aspects considered for creating the current scenario provide a framework of creating a new scenario (Figure 3).

Four design solutions have been generated to solve problems raised in the current situation: (1) an alphabet sound interpreter to connect a call without looking up the name of the receiver; (2) a speaker phone mode enabler to allow multiple people to talk together or for hands-free talking; (3) a camera

attached to the phone to show the caller's environmental surroundings to the receiver while talking; (4) a 'talk' button on the outside of the phone that can be pressed sight unseen. In Figure 3, problems and situations that justify a need for these changes are described. The new scenario clearly shows how the current situation is improved by the new solutions.

4. CONSTRUCTING ASPECT MODELS FROM DIFFERENT DISCIPLINES FOR SYSTEM IMPLEMENTATION

As mentioned above, the new scenario is encoded with the DIF in order to construct various kinds of models that support the implementation of the product. The following methods were selected as examples of various approaches for the implementation of a new product: object-oriented specifications [1] for software engineering, TKS (Task Knowledge Structure) for the human factors [4], and FBS (Function-Behavior-State) model [8] for mechanical/hardware engineering. These different models can be understood as examples of aspect models.

Use of scenarios for object-oriented approaches is common in software development. Known and observable tasks and objects in envisioned scenarios of use are easily transformed into software objects and their functions. The terms, "responsibilities" and "collaborations" that correspond to the functions of objects, define the structure of a software system effectively with the object-oriented concept [3]. This design approach is called "responsibility-driven design." The most popular sources of using scenario approaches for object-oriented software implementation are use cases. With the construction of specific scenarios from the use cases, CRC (Class-Responsibility-Collaboration) cards can be produced for the responsibility-driven design approach [9]. In the CRC cards, classes represent objects in scenarios, responsibilities represent the primary functions of a class, and collaborations describe other classes that a certain class should interact with in order to accomplish designated responsibilities.

Besides this approach, task analysis models such as Task Knowledge Structures (TKS) with Knowledge Analysis of Tasks (KAT) for software development from the human factor-oriented viewpoint especially for user interface design are important examples of the methods that support system design and implementation [4][5]. Task Knowledge Structures (TKS) provide a framework for describing tasks in terms of goals, subgoals, procedures, objects and actions. The TKS approach assumes that human activity associated with tasks follows a structured pattern determined by the organization, domain, task, and person's experience. It further assumes that people develop task knowledge from these structured patterns. The TKS is a framework for modeling this knowledge. It contains declarative knowledge about organization, context, goal structure, and procedural knowledge of how task activities are carried out. In this method, the TKS for the current tasks with current systems are used as a basis for constructing a "design task model," a new version of the task model for new design concepts. From the design task model, designers can produce interface models that describe interaction between users and system dialogs. Desirable inputs in specific states and outputs for these inputs can be designed and determined using this model.

The FBS model is an effective model for describing a system function structure to support its implementation. In this model, the meanings of functions and behaviors are clearly distinguished. Behavior can be defined objectively as transitions of physical states and therefore can be derived from physical states of an entity and its environment. However, function is related not only to physical behavior but also to the designers' perception of behavior. It is defined as "a description of behavior abstracted through recognition of behavior for utilization." [8] In this model, a function is represented as a combination of a body that carries meaning of the function, objective entities that a function occurs in, and function modifiers that detail the function body represented by adverb words like "precisely" or "firmly." Because the FBS model adapts the language structure for the description of this model, the transformation of scenarios into this model is viable.

These three distinct approaches can be applied by using one scenario source that will be encoded by the structured framework, a P-DIF example. Scenarios provide coherent support to various design activities such as requirements engineering, usability engineering, and system engineering with their common design language [3]. Encoding the scenarios with the shared structure and framework, the P-DIF, enables the integration of the outputs from a variety of design activities.

The P-DIF encoded scenario is used to construct the selected models. To develop the DIF-based framework for encoding, the identification of necessary variables is required for constructing each model. Table 1 shows the P-DIF components with a part of the encoded data from the new scenario. These data are used for constructing the selected models.

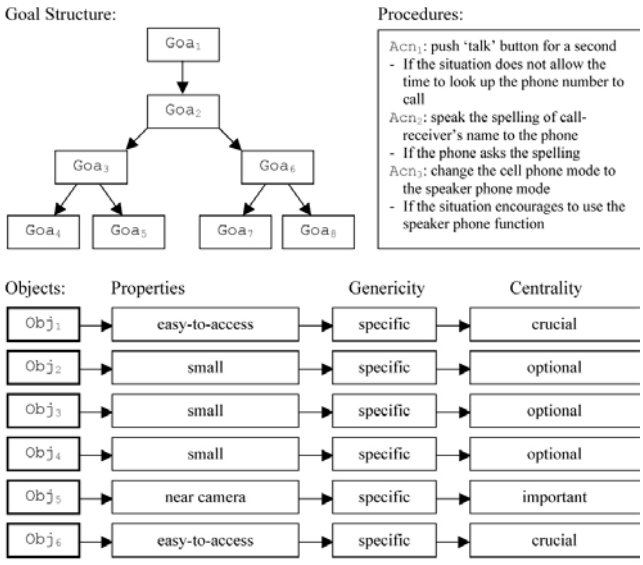

Figure 4: A TKS Model Generated from the New Scenario

In this case, user knowledge structured through the TKS framework (Figure 4) provides design guidelines. Genericity and centrality define the arrangement of objects for the product interface. For example, a specific but crucial object should be located in an easily accessible place while a specific and optional object should be located in a hidden place because the user does not always need the function related to that kind of an object. The speaker phone function-related objects are an example of this situation.

Unlike the TKS framework, the CRC cards describe how the internal mechanism is structured for implementation. One example of the CRC cards on Figure 5 shows that the alphabet interpreter can be enacted to perform its responsibilities by collaborating with the alphabet sound recognizer or the address book.

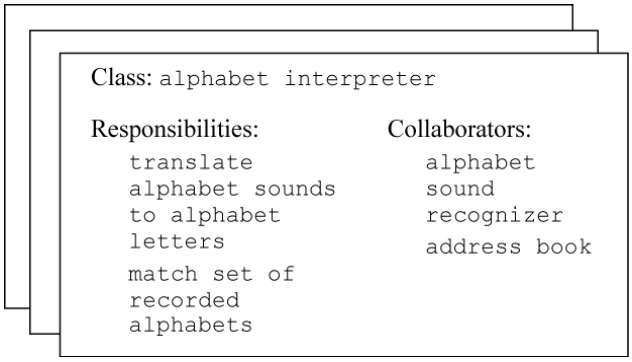

Figure 5: A CRC Card Example Created from the New Scenario

Whereas the CRC cards deal with the software aspect for implementation, the FBS model focuses on the implementation of the hardware aspect for the product (Figure 6). It identifies relationships between functions. This guides the hardware structure of the product by specifying necessary components to realize the functions.

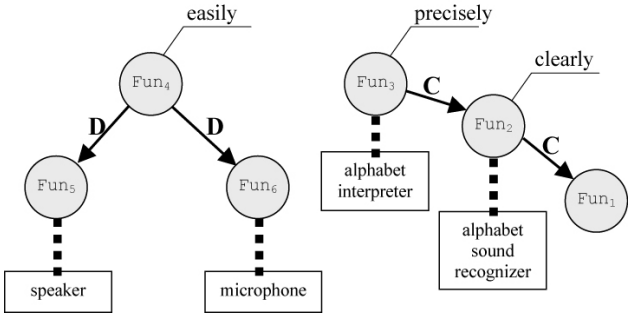

Figure 6: A FBS Model Generated from the New Scenario

The original P-DIF used for encoding the scenario enables the interpretation between these models by identifying the same DIPs. Obj_1 in Figure 4 is 'address book,' and this *entity* is also used as a Collaborator in the CRC cards (Figure 5). The Class related to this *entity*, 'alphabet interpreter' is used as an *entity* in the FBS model (Figure 6). By associating these three models through this channel, the access method to the address book is defined to be enabled by voice recognition which does not require any specific buttons for access. The FBS model shows which functions are related to each other for the alphabet interpreter. The Responsibilities defined in the CRC card for this object (Figure 5) are related to the Functions in the FBS model. The CRC card complements the information of Collaborators to support those Responsibilities or Functions in the function structure identified by the FBS model.

Through this interpretation, the relationship between the hardware components, such as the alphabet interpreter or the alphabet sound recognizer, and the software components, such as the address book, can be defined clearly in terms of the functions related to them. The relationship between the type of interface to access the address book and the function of the alphabet interpreter is also defined.

5. CONCLUSION AND FUTURE STUDIES

This case study demonstrates that encoding the scenarios of using a new product based on the structure of DIF provides an effective way of constructing various models of system implementation, and also enables the interrelated interpretation between those models to support the holistic development of the system. Through this research, the following were achieved:

- A holistic view of the problems through multi-aspect models in scenario construction; DIF provided the basic infra-structure to integrate the different aspects of use situations;
- Creation of an effective method for generating scenarios through use of aspect models based on the DIF structure;
- Evaluation of solution ideas in the early stage of the development process with scenarios, which can work as conceptual prototypes for simulating a use context;
- Generation of multiple aspect models from a scenario which can be inter-related in a structured way.

As an extension of the research, this mechanism can be used for other design activities, such as creating functional requirements and specifications for new solutions, as well as usability analysis with solution scenarios. For future development, what needs to be investigated are the issues of reconstructing scenarios by integrating different scenarios or of deconstructing a single scenario.

6. REFERENCES

[1] Booch, G., Rumbaugh, J., and Jacobson, I., **The Unified Modeling Language User Guide**, Reading, MA: Addison-Wesley, 1999.

[2] Breitman, K. and Leite J., "A Framework for Scenario Evolution," **Proceedings of the 1998 International Conference on Requirements Engineering (ICRE'98)**, the IEEE Computer Society, 1998.

[3] Carroll, J. M., **Making Use: Scenario-Based Design of Human-Computer Interactions**, Cambridge, MA: The MIT Press, 1998.

[4] Johnson, P., Johnson, H., and Wilson, S., "Rapid Prototyping of User Interfaces Driven by Task Models," **Scenario-Based Design: Envisioning Work and Technology in System Development**, New York: John Willey, 1995, pp. 209-246.

[5] Lim, K., and Long, J., **The MUSE method for Usability Engineering**, Glasgow: Cambridge University Press, 1994.

[6] Lim, Y. and Sato, K., "Development of Design Information Framework for Interactive Systems Design," **Proceedings of the 5th Asian Design Conference**, International Symposium on Design Science, Seoul, Korea, 2001.

[7] Mackay, W. E. and Fayard, A., "HCI, Natural Science and Design: A Framework for Trangulation Across Disciplines," **Conference Proceedings on Designing Interactive Systems**, ACM Press, Amsterdam, 1997, pp.223-234.

[8] Shimomura, Y. et al., "Representation of Design Object Based on the Functional Evolution Progress Model," **Proceedings of the 1995 Design Engineering Technical Conferences**, Vol. 2, ASME, 1995, pp.351-360.

[9] Wirfs-Brocks, R., "Designing Objects and Their Interactions: A Brief Look at Responsibility-Driven Design," **Scenario-Based Design: Envisioning Work and Technology in System Development**, New York: John Willey, 1995, pp. 337-360.