Abstract

We propose a method for recording and analyzing actual design processes, which is called Problem-Product-Knowledge (PPK) method. This method consists of two parts: a method for recording design activities and a framework for organizing the record. The recording method allows designers to record their activities easily, and the framework allows analysts to organize a record into a design process description and provides two different views of a design process: problem solving process and product generating process. These characteristics of the method help analyzing actual design processes.

1 Introduction

We have worked on developing new design techniques as a part of a project which aims at establishing a base technology for software design [2]. In the early stages of the project, we have thought that it is necessary to describe and analyze the actual design process first.

A number of studies on software process have been made during the past few years. Most of the studies are intended to describe software process formally based on mathematical modeling technique. Their basic approaches seem to be similar to that of process programming [1] from the following two viewpoints. First, they tend to emphasize on describing design procedure or behavior rather than know-how, which is essential knowledge to support the tasks. Second, they don't mention how to design/make process programs to be described in their models.

Generally speaking, designing a process program is a difficult and labor-intensive task because it requires analysis of actual software processes, which needs recording the processes in advance of the analysis. There are no established methods to do the task systematically.

In this paper we propose a method for recording and analyzing actual design processes, called Problem-Product-Knowledge (PPK) method. In this method, a design process is regarded as a hierarchical network of activities connected by three types of relations: problem, product, and knowl-

dge.
views for analysts.

Our design process description method proposed here satisfies the above four requirements. The method is composed of:

- a recording method (2.2) and
- a framework for organizing the record (2.3).

### 2.2 Recording method

A recording method we propose consists of two procedures:

- designer’s procedure to record design activities and
- analyst’s procedure to question the designers.

First, according to the procedure of figure 1, designers record their design activities. Second, according to the procedure of figure 2, analysts question designers on unclear parts of the record and add some information to the record.

As a result of the two procedures, time stamps and three kinds of data are recorded: problems [a, a'], design products [b, b'], and knowledge [c, c'].

Repeat 1-3 until a design is finished.

1. list problems to be solved [a]; number the activity to solve the problems; record starting time of the activity.
2. try to solve problems in any way possible; suggested steps include
   - (a) writing notes and drawing figures/charts [b].
   - (b) write down as much of what you were thinking while doing 2a as possible [c].
   - (c) list the notes and figures/charts referenced [b].
   - (d) list newly arisen problems [a'].
   - (e) record suspending time and restarting time , and write down the reason.
3. write down what have been done so far after solving the problems: record ending time.

![Figure 1: Recording procedure](image)

#### 2.3 Framework

Through the recording process described in 2.2, the three kinds of results: problems, products, and knowledge are obtained. These three kinds of information can be connected to each other through each problem-solving activity. Since the record is both raw and extensive, it is hard to understand and analyze the record as it is. There must be a description framework with which analysts can view the record from various angles, and with which they can organize it.

Then, we propose a description framework which represents the three kinds of information and their connection through problem solving activities, and we called the framework Problem-Product-Knowledge (PPK) model shown in figure 3.

Since three kinds of information which are outputs of an activity can be inputs of another activity, a process described in the PPK model is a network of activities and the surrounding information. Analysts connect all the recorded information with the framework into such a network.

![Figure 3: PPK model](image)
This describing method together with the recording method described in 2.2 is called the PPK method.

3 Analytical characteristics

The PPK model has two advantages in analyzing a design process.

1. Providing two views:
   Depending on purpose of analysis, the viewpoint of a design process can vary. For example, when the designers themselves or the persons maintaining the software look at the design process, it is important for them to know how the problems have been broken down and how decision has been made. On the other hand, the manager would want to know which activities have produced which results. A key characteristic of the PPK model is that it provides two separate views which correspond to the views discussed above, that of the problem solving process and that of the product generation process.

   (a) Problem solving process:
   By just extracting problems and activities from a design process description with the PPK model (PPK description), we can create a problem tree such as an example shown in figure 4. With this problem tree, the process of problem solving can be traced.

   (b) Product generation process:
   A design process can be seen as the process of generating design products from other products. As described in 2.2, notes and figures/charts are created during a problem solving activity by referencing already existing notes and figures/charts. Therefore, by extracting design products and activities from a PPK description, the design process can be traced as a product generation process (figure 5).

2. Allowing description organization:
   A PPK description can be drawn hierarchically like a data-flow diagram. Certain small activities can be grouped together to form a single large activity with a new activity name. In this way, the details of problems, products and knowledge can be hidden within one large activity. In figure 6, two activities are combined into an activity.

4 Practice of PPK method

We have made three experiments of describing design processes so far. Through the experiments we have figured out some points which are needed for using the PPK method.
practically. In this section, we show them with some examples.

4.1 Recording and Questioning

Figure 7 is a problem solution activity recorded in the method described in (2.2).

The original record the designer wrote down is on the left side of figure 7, and some information the analyst added through questioning are on the right side.

The recording procedure shown in figure 1 imposes a few restrictions on designers in order not to bother their thinking process. Nevertheless we found out that it is difficult for designers to record the following information.

1. A problem to be solved [a]
   Designers sometimes begin an activity without specifying any problems.

2. Feeling of achievement [c']
   Since designers tend to move their attention from one problem to another when a new problem comes up, they often change activities without writing down their feeling of achievement of activities.

3. Background knowledge [c]
   Designers seldom write down their own background knowledge while they are solving.

These information should be filled in with the questioning procedure shown in figure 2. The most important thing analysts should take into account when questioning is to write down what designers were thinking [c'] as much as possible. This is the key to questioning designers on these missing information.

For example, the left side of figure 7 (recorded by a designer) does not tell what is the motive of the activity and what the figure represents. These information were added by analysts on the right side of the record.

However, there are some information which are hard to be filled in later on, such as referenced products [b], raised problems [a'], and information about time. Analysts should ask designers to write down as much of these information as possible in advance.

4.2 Describing

When analysts organize the resulting record into a PPK description, they should take care of the following points:

1. Good naming
   The most important things in organizing the record are to give a proper short name to each activity and product, and to summarize each problem and knowledge. In general, the framework to understand a human activity is consists of seven factors: type of activity, subject, object, purpose, result, method and condition [9]. These factors correspond to the PPK model as below:

   type of activity $\rightarrow$ activity
   subject $\rightarrow$ none
   object $\rightarrow$ input and output product
   purpose $\rightarrow$ input problem
   result $\rightarrow$ output knowledge
   method $\rightarrow$ input knowledge
   condition $\rightarrow$ input knowledge

   Figure 8: Description of figure 7

To get an understandable process description, analysts should puts these factors into the description properly.

Figure 8 shows a design activity described in the PPK model from the record in figure 7. All names in the description are quoted from the record.

2. Semantic organization
   If letting a PPK description remain as a plain network, it is hard to grasp the whole context of the process. An advantage of the PPK model is the ability to organize the description hierarchically: organizing several detailed activities with three kinds of information by surrounding them with a circle into a more abstract activity. It allows analysts to understand the process macroscopically.
When analysts try to surround some activities and organize them into a more abstract activity, analysts must take care of the following points:

(a) whether an appropriate name exists for the new activity.

The primary condition to organize several activities is that there exists a simple activity name for them. We observed the output knowledge as a key to the organization since in many cases feeling of achievement is not just an output of the last activity, but that of the sequence of activities before.

(b) whether much information can be hidden in the new activity.

(c) whether incoming and outgoing information of the new activity can be minimized.

Organization of activities makes the description more understandable (figure 9).

4.3 An example

We recorded and described an actual design process. Data of the process are shown below.

Problem: Address roster management program (personal database management tool)

main issues... design of a query language and a list arrangement function

scale... 2K lines of C source code

specification... vague

Designer: 4 year experienced C programmer, who has knowledge of YACC.

Figure 9 shows a PPK description for the first half of this design process, and provides a problem solving view of the process, which allows us to trace designer's thinking process.

5 Possible alternatives

The PPK method consists of a recording method and a framework for organizing the record (PPK model). These two parts are independent. That is, we can select another recording method to describe a process in the PPK model. All we need for the recording method is that it can identify sequence of activities and relations among the activities (problem, product, and knowledge). In that case, adequateness of selected recording method depends on the purpose of analyzing the process. For example, if more precise analysis of designer's thinking process is needed, a popular recording method in protocol analysis field: letting them speak out all they think of and recording all they do by video camera, may be more preferable to our recording method because of less disturbance to the designers.

6 Design method extraction

In this section, we discuss how to improve a PPK design process description and how to extract a design method from the improved description. We take up the design process description of figure 9 as an example.

6.1 Key to improvement

A key to improving a PPK description is to observe how problems have risen in the process. From the three experiments of describing processes, we found out that there are four types of problem rising:

1. external factors
   Such as machine down, meetings and so on.

2. intuition
   While solving a problem, designers may hit on other problems.

3. breakdown of a problem
   Before solving a problem, designers may try to find smaller problems. In this case, he can predict raise of some sub-problems.

4. arrangement of a procedure
   In this type, designers may not only find sub-problems, but also arrange a procedure to solve them by considering priority and efficiency.

In the above list, a larger number implies more refined attitude of designers, except for 1. We also found out that most of the problems had risen as type 2 and some of them changed the process drastically. One of the factors which make a design process poor is backtracks caused by misarrangement of the order to solve the problems.

6.2 Improvement

To make a design process better, we first need to eliminate simple mistakes and problems caused by external factors from the process description. And then we need to replace intuitively risen problems with efficient procedures to solve sub-problems. 1-2 in the procedure of figure 10 are for the
1. eliminate problems caused by external factors (figure 9 (3)).
2. find an unnecessary multitude of activities due to some mistakes (figure 9 (1)(2)), and eliminate them except the first one. Add the reasons for the mistakes into input knowledge of the first one (figure 11 (1)(2)).
3. separate intuitively risen problems from activities where the problems rise.
4. rearrange the description as a chain of products and activities.
5. determine the order of activities based on the dependencies among them.

Figure 10: Procedure for improving a design process

former, and 3 - 5 are for the latter.

Applying the procedure to the process description of figure 9, we get figure 11, which provides a product generating view of the process.

6.3 Extraction

In general, a design method such as JSP [10] and SA/SD [11] consists of:

1. a procedure of all design activities,
2. definitions of all design products which are input and output of the activities,
3. ways to evaluate the products to decide the next activity.

Of course, some design methods describe the other information, however, we adopt the above definition for design methods here [12]. According to the definition, we discuss how to extract a design method from a PPK description.

Figure 12 shows a procedure for extracting a design method from a design process to which the procedure of figure 10 is applied. Applying the procedure to the improved process description of figure 11, we get a design method (table 1).

Figure 9: Design process of Address Roster Management Program
1. Write down activities in the order of the activity number.
2. Modify each activity semantically, noting the input problems and input knowledge.
3. Write down input/output products of each activity, and
4. Extract post-conditions of each activity from the output knowledge.

Figure 12: Procedure for extracting a design method

6.4 Evaluation
We consider that the design method (Table 1) is not enough because of:

1. Lacking of formality
Design products which designers generate in their own way often have no syntax and semantics. In this case, we cannot extract a way to evaluate the products. For example, since “BNF” is a language with a rigid syntax, we can check its correctness by syntax checking, while “Fragmentary system description” is just a list of requirements in a natural language, so we cannot evaluate it.

2. Ambiguity on the range of application
A design process reflects characteristics of the designed software. Therefore, application of the extracted method to another design issue must be limited. Understanding the range of application is needed before using it. For example, a condition “Design of a personal database management tool, whose specifications are vague” is really helpful for designers to use the design method of Table 1.

3. No value assurance
In spite of improving an original process with the procedure of Figure 9, we cannot assure its quality: if the designed software is poor, then the design process is useless.

To solve these problems and get meaningful design methods, the following approach is needed:

1. Let designers only use diagrams with well-defined syntax and semantics to create their design products. It is also important for the designers to understand them well.

2. Collect many design methods extracted from the same domain. These methods have the similar application and different conditions. If there are a lot of design methods with the similar application, designers may retrieve a method whose conditions fit on their own design issue, and then may reuse the method with small change. Furthermore, it may be possible that such collection of design methods causes higher level abstraction of them.

3. Select processes which have good quality.
### Design Procedure

<table>
<thead>
<tr>
<th>Design Procedure</th>
<th>Input Product</th>
<th>Output Product</th>
<th>Post-condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Requirement definition</td>
<td></td>
<td></td>
<td>Fragmentary system description</td>
</tr>
<tr>
<td>To make your requirements clear, list functions, user interfaces, and constraints.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. System analysis</td>
<td></td>
<td></td>
<td>Fragmentary system description</td>
</tr>
<tr>
<td>To develop your system image, do system analysis with SA method, taking account of options of command.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Data definition</td>
<td></td>
<td></td>
<td>Dataflow diagram</td>
</tr>
<tr>
<td>Design the data listed in 2.</td>
<td></td>
<td></td>
<td>I/O scenario</td>
</tr>
<tr>
<td>User input and screen output $\rightarrow$ 3</td>
<td></td>
<td></td>
<td>until all data are clear</td>
</tr>
<tr>
<td>List data $\rightarrow$ 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>You can do 3 and 5 concurrently.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Development of an image of user I/O</td>
<td></td>
<td></td>
<td>Fragmentary system description</td>
</tr>
<tr>
<td>To make user input and screen output clear, develop input/output scenario.</td>
<td></td>
<td></td>
<td>I/O scenario</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>until you can get an image of the language for data retrieval.</td>
</tr>
<tr>
<td>4. Brief design of the language</td>
<td></td>
<td></td>
<td>I/O scenario</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BNF, Result of parsing</td>
</tr>
<tr>
<td>4.1 Write a BNF</td>
<td></td>
<td></td>
<td>I/O scenario</td>
</tr>
<tr>
<td>Using knowledge of LALR(1), write a BNF</td>
<td></td>
<td></td>
<td>BNF</td>
</tr>
<tr>
<td>of the language</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2 Choose an example, parse it</td>
<td></td>
<td></td>
<td>I/O scenario</td>
</tr>
<tr>
<td>To check if the BNF is correct and to develop an image of the action part, choose an example in I/O scenario and parse it.</td>
<td></td>
<td></td>
<td>Result of parsing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>until you can get an image of the action parts</td>
</tr>
<tr>
<td>5. Development of list operation routines</td>
<td></td>
<td></td>
<td>List operation routines</td>
</tr>
<tr>
<td>With design method of abstract data type, design the list data.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1 Structure variable definition</td>
<td></td>
<td></td>
<td>Structure variable</td>
</tr>
<tr>
<td>Based on the image of the list as an abstract data type, develop a list structure variable. In that time, take account of how to allocate attribute.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2 List operation routine definition</td>
<td></td>
<td></td>
<td>Structure variable</td>
</tr>
<tr>
<td>To define the list operations, code and debug list operation routines.</td>
<td></td>
<td></td>
<td>List operation routines</td>
</tr>
</tbody>
</table>

Table 1: Design method extracted from figure 11

### Related work

In our research project [2], three types of language are examined to describe design process: HFSP [3], PDL [4], and Pshell [5]. Both HFSP and PDL are functional and hierarchical languages, and each of them has an interpreter, which can execute the process description to support designer's activities directly. Pshell adopts rule-based paradigm and emphasizes on dynamic execution of the description. Another type of process modeling is proposed by W. Humphrey and M. Kellner [6], which enables to describe precise and detailed behavior of software process in order to control process scheduling.

Each of them has a different mathematical background to address different issues. But, they are not a process model enough to describe know-how, and don't have a systematic method for making the description.

In looking at ways of recording processes for design, reference can be made to the gIBIS proposed by the MCC Software Technology Program [7] [8]. gIBIS is a tool to support issue-based decision making among design team, and is based on IBIS model, in which a design process is treated as a conversation based on each person's viewpoints of his design problems, with issues, artifacts, alternatives and justifications (an actual record of a software design is reported in [8], through a test use of gIBIS). The gIBIS's model for recording design processes is not a model for analyzing the
record. For example, it is not suitable for organizing a batch of records in a hierarchical manner.

8 Conclusion

A PPK description is quit simple and understandable because the model is based on general framework to understand a human activity. The PPK model can be considered as a tool which analysts can construct a tree (or dag) of designer's activity in a bottom-up manner. This tree/dag includes abstraction of both problem-solving-process and product-generating-process. Such an abstraction greatly makes the process understandable. Although the PPK method is still informal, we believe that it is useful as a first step of modeling actual design processes.

In the later part of this paper, we have discussed extraction of a design method from a design process description. This demonstrates a capability of the PPK model.

We are planning further experiments to describe a lot of actual design processes. We believe that such experiments are the best way to develop new technologies for evaluation, reuse, and support of design processes as well as to improve the method itself.

References


