Comparison of Two Approaches for Automatic Construction of Web Applications: Annotation Approach and Diagram Approach

Mitsuhisa Taguchi, Kornkamol Jamroendararasame,
Kazuhiro Asami, and Takehiro Tokuda

Department of Computer Science, Tokyo Institute of Technology
Meguro, Tokyo 152-8552, Japan
{mtaguchi, konkamol, asami, tokuda}@tt.cs.titech.ac.jp

Abstract. In order to support development of consistent and secure Web applications, we have designed a number of Web application generators. These generators can be classified into two types of approaches: annotation approach and diagram approach. In this paper, we try to make the roles of these generators clear, and compare two approaches in terms of target applications, developing processes and target users. While both approaches are powerful and flexible enough to construct typical Web applications efficiently, we may select the most appropriate generator according to the characteristics of the application and the developing process.

1 Introduction

Today, Web applications such as database query systems and transaction systems are widely used especially on the Internet. The development of such applications, however, requires much cost and experience of developers because of the complexity of security checks and session management, which are unique to Web applications. In order to support development of consistent and secure Web applications, we have designed a number of Web application generators which generate source codes necessary to execute the application [1,2,3,4,5,6, 7,8,9]. Web application generators encapsulate the complexity unique to Web applications and make developers concentrate on the business logic of the application.

Our generators can be classified into two types of approaches. The first is annotation approach, which concentrates on input data and embedded values on each Web page [1,2]. Developers first compose Web page templates and give declarative annotations to them. And then the generator generates procedural program codes from Web page templates with annotations. The second is diagram approach, which concentrates on data-flow relationships in the application [3,4,5,6,7,8,9]. Developers first compose diagrams which describe overall data-flow relationships among Web components such as Web page templates, server

N. Koch, P. Fraternali, and M. Wirsing (Eds.): ICWE 2004, LNCS 3140, pp. 230–243, 2004.

side programs and databases. After developers select appropriate program templates and components, a generator can generate executable program codes from the diagrams.

Each approach has assumptions and roles in developing processes. Developers can select the most appropriate generator according to the characteristics of the application and the developing process. In this paper, we try to make the roles of these generators clear, and compare the two approaches in terms of target applications, developing processes and target users. We also give discussions on our future work based on this comparison.

The organization of the rest of this paper is as follows. In section 2 and section 3, we describe annotation approach and diagram approach, and give examples of our generator systems based on each approach. In section 4, we compare two approaches in terms of target applications, developing processes and target users. In section 5 we give related work. We finally give future work and concluding remarks in section 6.

2 Annotation Approach

2.1 Basic Idea of Annotation Approach

From the viewpoint of user interfaces, we can consider general Web applications as transitions between Web pages just like ordinary Web pages that have no server side programs. In most Web applications, the greater part of each Web page template is static, and a part of the template is dynamic where actual values are embedded by server side programs. Based on this idea, we present annotation approach to automatic construction of Web applications. In annotation approach, we describe dynamic part of Web page templates as declarative annotations. More precisely, the following steps are taken in the generation method.

1. We first construct Web page templates for intended Web applications. Web page composers and other support tools may be used to visually compose Web page templates efficiently. Dynamic part of the templates, where values are embedded at run time, may be described as special characters to distinguish from static part.
2. We give annotations to dynamic part of Web page templates. Our annotations are declaration of data processing which is executed on the server side. Their tasks are mainly related to data-flow relationships among Web components as follows.
   - For input data checking, such as checking acceptable types and length of input values and constraints between them.
   - For session management, such as checking the beginning and the end of the session.
   - For database handling, such as the access to database management systems using queres.
   - For communications with external programs, such as invocation of Web services, EJB components and other applications.

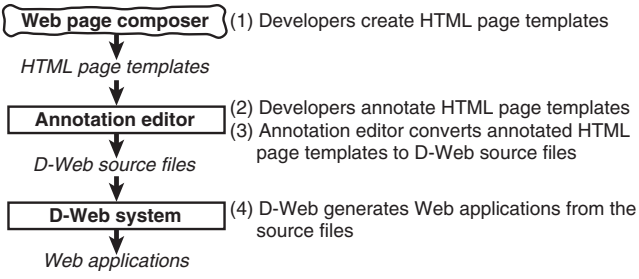

Fig. 1. The architecture of A-Web system

3. From Web page templates with annotations, a Web application generator automatically generates source codes of server side programs. Generated programs have consistency and the standard level of security because the generator checks transitions and parameters between Web page templates, and generates additional codes to prevent inconsistency.

2.2 Generator: A-Web System

Based on annotation approach, we designed and implemented a prototype generator called A-Web system which generates Web applications from Web page templates [1,2]. While our current prototype can generate only CGI programs, we may be able to generate Web applications based on other architecture by replacing part of the generator because annotation approach itself is independent of specific architecture.

The architecture of A-Web system is shown in Fig.1. A-Web system consists of two parts: an annotation editor and a Web application generator called D-Web system.

Web page templates. Before using A-Web system, Web page templates should be prepared as input of the system. We can compose Web page templates using visual composers and other support tools because A-Web system requires ordinary XHTML documents. Dynamic part of the templates should be represented by special characters ${*scope.valiable*}. *Variable* is a name to distinguish from other variables among the application and should be unique in each *scope*. Because the special characters are ordinary strings, not extension of HTML tags, common tools can deal with these templates. Thus it is unnecessary for us to modify the source codes directly.

Fig.2 shows an example of Web page templates of a simple member registration system. This application first requires users to input id and password which the user chooses, a name, an email address and so on at a Web page 'Registration'. If the id is already registered or the input data don't satisfy specified conditions, a Web page 'Error' is generated. Otherwise a Web page

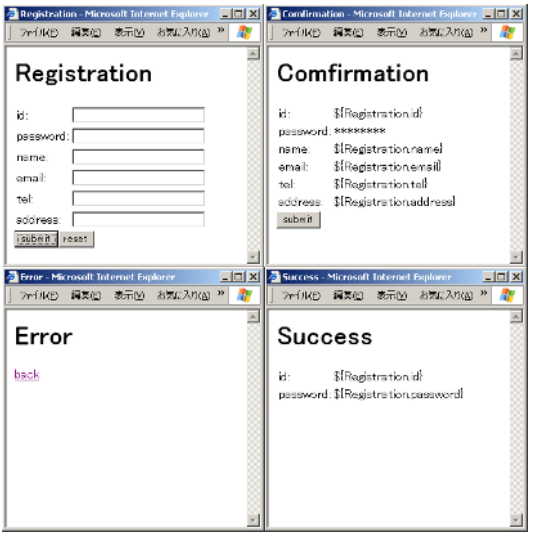

Fig. 2. An example of Web page templates

'Confirmation' is generated. After the confirmation, the registration becomes definite.

An annotation editor. An annotation editor is a part of A-Web system, which is an editor to annotate Web page templates and convert them to D-Web source documents. When the editor gets Web page templates, it analyzes the templates, points out where we can give annotations and allows us to give annotations visually. In our prototype, we introduce five types of annotations: input check, constraint, session, SQL and SOAP.

An input check annotation defines conditions for acceptable users' input. If the input is strings, we can define the length and types of acceptable characters. If the input is numbers, we can define a range of acceptable numbers.

A constraint annotation defines relations which must be satisfied among input data.

A session annotation defines the behavior of session management for each template. When a user accesses a Web page with a session annotation 'begin', the program starts the new session. In the case of 'check', the program checks whether the session is valid or not. In the case of 'end', the program terminates the session.

A SQL annotation describes SQL statements to access database management systems. We can deal with database transactions.

A SOAP annotation describes statements to invoke external Web services using SOAP protocol.

Fig.3 shows an annotation editor in A-Web system. It analyzes Web page templates and adds hyperlinks to special annotation pages automatically.

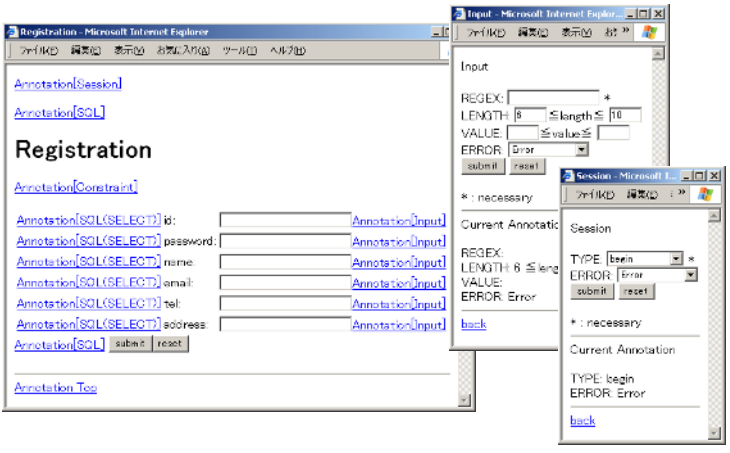

Fig. 3. An annotation editor of A-Web system

```
<html>
<head>
  <title>Registration</title>
  <session type="begin" error="Error.html"/>
</head>
<body>
  <h1>Registration</h1>
  <form action="Confirmation.html" method="Post">
    id:<input type="text" name="id" length="[6,10]"/>
      ---- omit following parts ----
</body></html>
```

Fig. 4. An example of D-Web source codes

Fig.3 shows 'Registration' page, a session annotation page and an input annotation page. Fig.4 shows an example of D-Web source codes, which are generated by the annotation editor from a Web page template 'Registration' and its annotations. D-Web source codes have a number of extended tags.

D-Web system. D-Web system is a part of A-Web system, which gets D-Web source documents and generates source codes of Web applications. In our prototype, D-Web system generates XHTML documents and CGI programs written in Perl. To keep consistency of generated applications, D-Web system analyzes all variables in Web page templates and transitions between Web pages, and gives methods to pass the values between Web pages correctly. To keep the standard level of security, all generated programs have additional codes as follows.

1. After receiving input data, each program checks whether the user comes from correct Web page.

2. Then each program checks whether user's session id is valid.
3. Then each program checks whether input data are correct according to input check annotations.

If there is at least one error, the user's request is redirected to an error page.

3 Diagram Approach

3.1 Basic Idea of Diagram Approach

From the viewpoint of data-flow relationships among Web components such as Web pages and programs, we can consider Web applications as applications based on pipes and filters architecture. A filter corresponds to a server side program and a pipe corresponds to a Web page which passes data between programs. Based on this idea, we present diagram approach to automatic construction of Web applications. In diagram approach, we first compose diagrams describing overall behavior of the application and select general-purpose templates and components to generate executable applications. More precisely, the following steps are generally taken in the generation method using this approach.

1. We first compose directed graphs whose nodes represent Web components such as Web page templates, server side programs and databases, and whose edges represent data-flow relationships among the components. Most of our generators use diagrams called *Web transition diagrams* which we designed for the above purpose.
2. All generators based on diagram approach have predefined program templates and components which are independent of specific domains of applications, for example, for purposes of database manipulations, sending electronic mails. Referring specifications of these programs, we give correspondence between nodes in diagrams and the programs, and give values of parameters of them.
3. From the descriptions of diagrams and values of parameters, a generator automatically generates Web page templates and source codes of server side programs. Generated programs have consistency and the standard level of security, because the generator checks consistency of the diagrams, checks correspondence between diagrams and predefined programs, and then generates additional codes to prevent inconsistency.

3.2 Generator: T-Web System

Based on diagram approach, we designed and implemented a prototype generator called T-Web system which generates Web applications from directed graphs called Web transition diagrams [3,4,5,6,7,8,9]. While we have prototypes to generate CGI programs, JSP/Servlet programs and ASP programs respectively, we may be able to generate applications based on other architecture by replacing part of the generators because diagram approach itself is independent of specific architecture of Web applications.

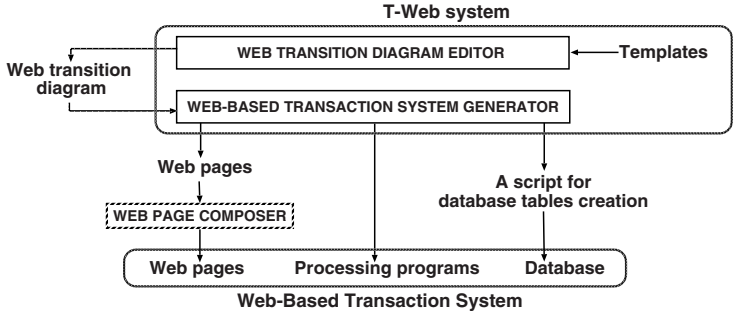

Fig. 5. The architecture of T-Web system

In this section, we explain the basic architecture of T-Web system on the basis of the generator for JSP/Servlet-based applications. The architecture of T-Web system is shown in Fig.5. T-Web system consists of two parts: a Web transition diagram editor and a Web application generator.

Web transition diagrams. In T-Web system, we describe overall behavior of intended application as directed graphs called Web transition diagrams. Basically, Web transition diagrams consist of four types of nodes and two types of links as follows.

A fixed Web page node is a static Web page which can be reached by a certain URL. Its notation is a rectangle with its name, whose line is thick. It may have a number of page elements such as hyperlinks and input fields inside the rectangle.

A output Web page node is a dynamic Web page which is generated by a server side program. Its notation is a rectangle with its name, whose line is thin. Like a fixed Web page node, it may have a number of page elements.

A processing node is a server side program which is activated by users' requests to perform processing. Its notation is an oval with its name.

A database node is a relational database table in a database server. Its notation is a cylinder with its name. The schema of the table is represented by a list of names between '{' and '}'.

A page transition link is a hyperlink relationship between Web pages. Its notation is a directed line.

A data-flow link is a data-flow relationship among Web components such as Web pages, programs and database tables. Its notation is a directed line with a blocked line.

Each generator may have some extension of Web transition diagrams according to the target architecture. Fig.6 shows an example of Web transition diagrams of a simple member registration system which is the same application as given in section 2.

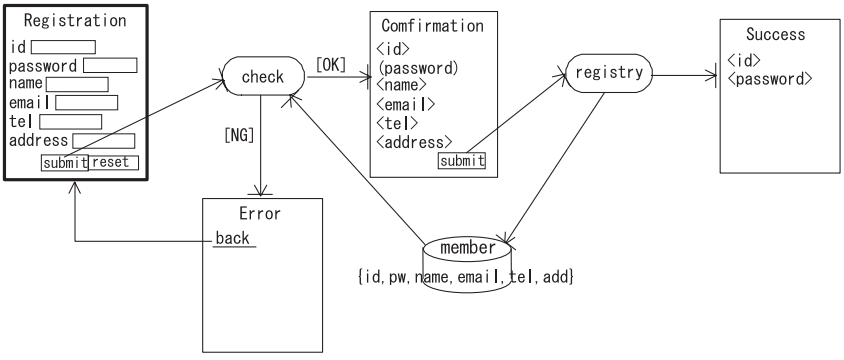

Fig. 6. An example of Web transition diagrams

A Web transition diagram editor. A Web transition diagram editor is a part of T-Web system, which is an editor to support composition of consistent Web transition diagrams. It allows users to do all operations visually. We compose Web transition diagrams as follows.

1. We start up the editor and the editor receives available program templates and components automatically.
2. We draw Web transition diagrams by selecting a node from a list, putting it on a drawing field and arrange nodes. A links is given by selecting the target node from a list so that we can give only syntactically correct links.
3. We specify details of each node using a window called *a property window*. When we specify details of a processing node, we should select the most appropriate program template from a list.

Fig.7 shows a Web transition diagram editor of T-Web system.

A Web application generator. A Web application generator is a part of T-Web system, which gets intermediate documents a Web transition diagram editor generates, and generates source codes of server side programs and Web page templates. Our prototype system shown in Fig.5 generates JSP documents written in HTML and servlet source codes written in Java.

Generators have general-purpose program templates which are independent of specific domains of applications so that we can widely reuse them. The above prototype has 16 program templates which are mainly for database manipulations, session management and sending electronic mails. Fig.8 shows an example of program templates in T-Web system. In our templates, a word between special characters '/*' and '*/' represents a parameter. Special characters '/**' and '**/' mean a repetition part. As the generator rewrites above parameters to corresponding values, the program template becomes a complete program.

To keep the standard level of security, all generated programs have additional codes as follows.

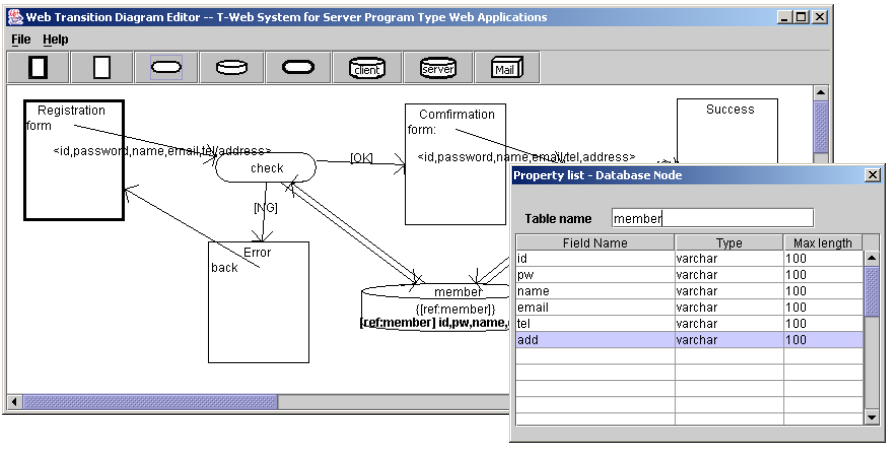

Fig. 7. A Web transition diagram editor of T-Web system

1. After activated by a user's request, each program checks whether user's session id is valid.
2. Then each program checks whether input data are correct according to specifications we give.
3. After processing of business logic, each program checks whether output data are correct according to database specifications. Especially when the program updates database tables, it checks consistency between output data and specifications of the database table.

If there is at least one error, the user's request is redirected to an error page.

4 Comparison of Two Approaches

We compare annotation approach and diagram approach in terms of target applications, developing processes and target users. We discuss target applications dividing into three viewpoints: application domains, flexibility and scalability. We discuss them according to the following criteria.

1. **Backgrounds** show why the viewpoint is important to select a generator.
2. **Characteristics** show the advantage and the disadvantage of two approaches and compare them.
3. **Examples** show our practical experience and other points to notice.

4.1 Application Domains

Backgrounds. Web applications can be generally classified into two types according to their functions: data-centric Web applications and control-centric

```
public class /*CLASSNAME*/ extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                                  throws ServletException, IOException {
    String myHttpsURL = "/*HTTPSBASEURI*/"+"/*PROGRAMURI*/"
                          +"/*PACKAGE.*/"+"/*CLASSNAME*/";
    String dbName = "/*DBNAME*/";
    String tbName = "/*TABLE1*/";
    String[] dbColNames = {/*"FIELDNAME1"*/};
    /**String PARAMETER = "";**/
    Connection con = TWeb.doConnect(dbName);
      --- omit following parts ---
  }
}
```

Fig. 8. An example of program templates

Web applications. Data-centric Web applications have side effect computations such as operations to update databases, while control-centric Web applications concentrate on execution of complex business logic.

Characteristics. Both our annotation approach and diagram approach concentrate on data-centric Web applications. Our Web application generators aim to encapsulate complex tasks unique to Web applications such as session management and security checks, which are greater problems in data-centric Web applications than in control-centric ones. In general, both approaches can deal with applications having functions as follows: database manipulations, invocation of external programs and sending/receiving electronic mails. On the other hand, both approaches are not good at dealing with complex page transitions which depend on the results of programs.

Annotation approach is useful to generate Web applications whose structure of Web page templates is complex, because we can use general Web page composers and authoring tools to compose flexible page layouts before the generation. Diagram approach is not good at developing such applications, especially clickable maps and multiple frames, because it is complicated work to connect data-flow relationships to the above components after the generation. On the other hand, diagram approach doesn't care a ratio of static parts and dynamic parts in each Web page, while annotation approach is not good at developing Web applications whose ratio of dynamic parts is high.

Examples. We are successful in developing typical data-centric Web applications using A-Web system and T-Web system respectively: shopping cart systems, guestbook systems, glossary systems, schedule organizing systems, member registration systems and reservation systems. The most appropriate generator mainly depends on the complexity of appearance of Web pages but not on application domains. When we want to develop more complicated Web applications, general software techniques such as object-oriented analysis/design and the extension of them may be helpful[11].

4.2 Flexibility

Backgrounds. When we need new business logic to generate an intended Web application, it is a big problem how to add new program templates and components into generators. It is also important what types of architectures it can deal with, because we often have constraints on the execution environment of the application.

Characteristics. In annotation approach, we can set new program components and give annotations to invoke the program. In diagram approach, we have only to add new program templates, program components and documents of their specifications, because the generators usually load specifications of all program templates and components automatically when it starts up. Diagram approach seems to be more flexible than annotation approach, because program templates encapsulate invocation of external programs and it is easier to add new program templates. When we want to generate Web applications based on various architectures, we can replace a part of generators in both approaches. Especially in diagram approach, we may have only to replace program templates to achieve the above goal.

Examples. Using our generators, typical data-centric Web applications are generated without new program templates and components. However, we possibly need new business logic in the case of a complex shopping Web site. In that case, it requires high programming skills and much cost to test them to produce new program templates and components. We have implemented generators and made sure that our approaches can deal with CGI, ASP and JSP/servlet architectures. When we extend a generator so that it can deal with new architecture, it requires very high programming skills. In that case, diagram approach may be easier than annotation approach.

4.3 Scalability

Backgrounds. The scale of the intended application is also an important problem as well as the complexity of Web page templates.

Characteristics. In annotation approach, we can basically concentrate on transitions between two Web pages, because the generator automatically gives a method to pass values of parameters from the origin page to the pages where they are used. Thus, even if the scale of an intended Web application becomes larger, the task of generation doesn't become so complicated. In diagram approach, however, the larger the scale of an intended Web application becomes, the more complex the management of diagrams becomes. While we can make diagrams nested, it is still a big program how to manage specifications of databases which are used in several diagrams.

Examples. Generators based on diagram approach can deal with no more than 15 Web page templates easily. While 15 Web page templates are enough for typical Web applications, a complex shopping Web site may exceed the number of templates. It may be helpful to use site diagrams which represent the structure of the Web site and static relations among Web page templates.

4.4 Developing Processes

Backgrounds. Web applications are characterized by three major design dimensions: *structure*, *navigation* and *presentation*[10]. We discuss effective developing processes in each approach taking notice of the order of these three design activities.

Characteristics. In annotation approach, we generally take the following developing process.

1. We first describe requirement specifications and analyze them.
2. We decide what Web pages are needed in the application, and design data structure (as a structure model) which the application deals with. From the above two models, we design Web page templates (as a presentation model) and implement them. We also design data-flow relationships among the application (as a navigation model) based on the above two models.
3. From the implementation of Web page templates and the navigation model, we generate server side programs using the generator.

In diagram approach, we generally take the following developing process.

1. We first describe requirement specifications and analyze them.
2. We decide what Web pages are needed, design data structure (as a structure model) and design data-flow relationships among the application (as a navigation model). From the above three models, we compose diagrams such as Web transition diagrams.
3. From the diagrams, we generate server side programs and prototypes of Web page templates using the generator. And then, we design Web pages (as a presentation model) and revise generated templates.

If we want to revise the appearance of Web pages after the generation easily, we should use XSLT and CSS to compose Web page templates in both approaches.

Examples. Annotation approach has the advantage of composing prototypes of Web page templates early in the process and reusing them for the final product. We can take an iterative and incremental process composing Web page templates. On the other hand, it is the disadvantage that it's hard to implement navigations as a prototype before composing Web page templates. Diagram approach has the advantage of designing and implementing structure and navigation iteratively and incrementally. On the other hand, it is the disadvantage that we cannot start implementation of Web page templates before the generation.

4.5 Target Users

Backgrounds. The knowledge a generator requires is important for developers to use the generator.

Characteristics. In annotation approach, Web page designers compose page templates before the generation. Users of the generator don't need knowledge of markup languages, because an annotation editor points out where

and what types of annotations we can give. The users also don't need knowledge of architecture of Web applications. On the other hand, the annotation based generator requires knowledge of data types, regular expression and the concept of session management. To develop advanced Web applications, it may require knowledge of APIs to invoke external programs. In diagram approach, users don't need knowledge of markup languages and programming languages, because program templates and components encapsulate them. The generator requires knowledge of Web interfaces, the concept of session management and basic architecture of Web applications. It also requires a skill in selecting and using general-purpose programs.

Examples. Annotation approach is effective for programmers who have basic knowledge of software development. Diagram approach may be easier for inexperienced programmers, because most typical data-centric Web applications have database manipulations and input checks, which are encapsulated by the generator.

5 Related Work

Currently there are many languages and tools to support development of Web applications. One of the widely used server side technology is so-called server side scripting such as ASP, JSP and PHP. We give fragments of program codes into Web page documents to describe dynamic parts of the Web page. However, we still have to give processing steps of session management and security checks in detail, because most of the program codes are procedural. Similarly, most support tools to generate a part of the above program codes require procedural programming. Thus, the encapsulation of such techniques is not enough for inexperienced programmers.

We may observe that there are many types of diagrams used for design or construction of Web applications. The extension of UML diagrams is one possible approach [11]. These diagrams are available to describe not only data-centric Web applications but also control-centric Web applications. However, it is not easy to use such diagrams for automatic construction because the descriptions are too detailed for inexperienced programmers to understand them. The aims are different in such techniques and our software generators.

One of the systematic construction methods of Web applications is object-oriented hyper media design method [12]. This method is to produce interfaces of Web applications from abstract data types using three types of diagrams. It presents the developing process from requirements to the products and diagrams to use. While it is useful under specific conditions, it is not so flexible because the starting point of the developing process is always abstract data types.

6 Conclusion

We have presented two approaches to automatic construction of consistent and secure Web applications. We designed and implemented Web application generators based on each approach. In this paper, we showed the roles of these

generators, and compared two approaches in terms of target applications, developing processes and target users. Both approaches are powerful and flexible enough to construct typical data-centric Web applications efficiently. Especially annotation approach has the advantage of developing applications whose Web page templates have flexible page layouts. On the other hand, diagram approach has the advantage of rapid and iterative/incremental development.

As our future work, we may try another approach which is based on the combination of two approaches. In this approach, we can develop programs and Web page templates concurrently. Such generator makes developing processes more flexible.

References

1. K. Asami, and T. Tokuda. Generation of Web Applications from HTML Page Templates with Annotations. *Proceedings of the IASTED International Conference, APPLIED INFORMATICS*, pp.295-300, 2002.
2. K. Asami and T. Tokuda. Generation of Web Applications from Annotation-Based Definitions. *Proc. of Engineering Information Systems in the Internet Context*, pp.69-79, 2002.
3. K. Jamroendararasame, T. Suzuki and T. Tokuda. A Generator of Web-based Transaction Systems Using Web Transition Diagrams. *Proc. 17th Japan Society for Software Science and Technology*, 2000.
4. K. Jamroendararasame, T. Matsuzaki, T. Suzuki, T. Tokuda. Generation of Secure Web Applications from Web Transition Diagrams. *Proc. of the IASTED International Symposia Applied Informatics*, pp.496-501, 2001.
5. K. Jamroendararasame, T. Matsuzaki, T. Suzuki, T. Tokuda. Two Generators of Secure Web-Based Transacion Systems. *Proc. of the 11th European-Japanese Conference on Information Modelling and Knowledge Bases*, pp.348-362, 2001.
6. K. Jamroendararasame, T. Suzuki, T. Tokuda. JSP/Servlet-Based Web Application Generator. *18th Conference Proceedings Japan Society for Software Science and Technology*, 2C–1, 2001.
7. K. Jamroendararasame, T. Suzuki, and T. Tokuda. A Visual Approach to Development of Web Services Providers/Requestors. *Proc. of the 2003 IEEE Symposium on Visual and Multimedia Software Engineering*, pp.251-253, 2003.
8. M. Taguchi, T. Susuki, and T. Tokuda. Generation of Server Page Type Web Applications from Diagrams. *Proc. of the 12th Conference on Information Modelling and Knowledge Bases*, pp.117-130m 2002.
9. M. Taguchi, T. Suzuki, and T. Tokuda. A Visual Approach for Generating Server Page Type Web Applications Based on Template Method. *Proc. of the 2003 IEEE Symposium on Visual and Multimedia Software Engineering*, pp.248-250, 2003.
10. Piero Fraternali. Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. *ACM Computing Surveys Vol.31 No.3*, pp.227-263, 1999.
11. J. Conallen. Modeling Web Application Architectures with UML. *Communications of the ACM Vol.42 No.10*, pp.63-70, 1999.
12. G. Rossi and D. Schwabe. Designing Computational Hypermedia Applications. *Journal of Digital Information, Vol. 1, No. 4*, 1999.