# Developing Enterprise Web Applications Using the Story Driven Modeling Approach

Christoph Eickhoff, Nina Geiger, Marcel Hahn, and Albert Zündorf

University of Kassel, Software Engineering,
Department of Computer Science and Electrical Engineering,
Wilhelmshöher Allee 73,
34121 Kassel, Germany
{cei,nina.geiger,hahn,zuendorf}@cs.uni-kassel.de
http://se.eecs.uni-kassel.de

**Abstract.** Today's browsers, tools and Internet connections enable the growth of Enterprise Web Applications. These applications are no longer page-based and designed using HTML code. Enterprise Web Applications bring the capabilities and concepts of traditional desktop applications to the browser. We are used to the development of desktop applications for years and have defined our own process to enable the full model-driven development of applications without source code. Using this process and its tools, we are able to define not only data models for traditional applications and generate code out of it. Combined with the story-driven modeling approach, we are able to design the logic of applications using models and generate fully functional code. To use our knowledge and tools as well as our usual process for the development of Enterprise Web Applications, we investigated our process and adapted it to the new needs. As result we propose a new development process that combines the needs of complex software development with the implementation of web user interfaces and control flows between these user interfaces. The process is a guideline to use models and tools for the development of complex Enterprise Web Applications including data model, behaviour and user interface.

## 1 Introduction

We have developed traditional Java applications for years. Our main research area has been the model- and story-driven development of such applications. We have investigated ways to do the complete development of applications with our own development process and own tools: The Fujaba Process [2], [3] and Fujaba Toolsuite [9]. However, over the last years the type of applications changed. We more and more faced the challenge of developing so called Enterprise Web Applications. What we exactly mean by this term is further defined in section 2. Since we have gained expertise in the modeling of applications over years, the question we asked ourselves was: "Is it possible to develop web applications without the need to write any sourcecode, too?". We started by investigating our development of traditonal Java applications and apparently faced the differences

A. Harth and N. Koch (Eds.): ICWE 2011 Workshops, LNCS 7059, pp. 196–210, 2011.

between traditional software and web applications. First of all, web applications will mostly be divided into a client part and a server part. These two parts communicate using Remote Procedure Calls or other request technologies known in the web domain. These topics were not clearly addressed in the development process we used until now. Additionally, there is no full modeling and code generation support for these techniques inside the Fujaba Tool Suite, yet. Another point missing in the traditional Fujaba Process is the design of Graphical User Interfaces (GUIs) and the binding of application data to these interfaces (databinding). The last point had resided on our todo list even before web applications turned out to be the new research point. We thus decided to enhance our processes and tool integration to be able to do full development of Enterprise Web Applications, including distributed application parts and user interfaces. We therefore started to adopt and enrich the traditional Fujaba Process to be able to support all these new requirements. Also we tried to support the process with tool integration of the new requirements. We propose the use of the Google Web Toolkit (GWT)[1] as user interface and client side logic library. GWT automatically generates browser specific JavaScript code from Java Code. The Java code can easily be generated using the Fujaba Toolsuite code generation mechanisms. A first step towards modeling of databinding and support for server calls with GWT was already introduced with the Fujaba *Action Charts*, [5], in 2010. The design of user interfaces as well in source code as in story-driven modeling is a painstaking task. We therefore propose the use of the GWTDesigner[2] which is a visual GUI builder generating Java code. To enable the completely model driven development of web applications, we have to close the gap between Java code for the visual components and the diagrams modeling the rest of the application. We propose to use UML Lab [3] and its reverse engineering technologies to derive a structural representation of the GUI. This is sufficient to enable the modeling of client side behaviour, databinding, GUI listeners and server calls using story-diagrams and *Action Charts*. While the main points of the Web Fujaba Process have already been defined and will be presented in this paper, there are still some weak points. We currently do not explicitly state how the databinding will be incorporated into the process. Also, we have automatic test generation for the server parts, but not for the logical parts residing in the client and the user interface. These points will be handled in future revisions of the process. The remainder of this paper is structured as follows: Section 2 defines the class of targeted applications. After having defined the application class, we introduce the story-driven modeling approach, our adaptations needen for web applications, as well as a running example in section 3. As a result of our research, section 4 answers the question raised in the Introduction and defines the Web Fujaba Process and the associated toolchain. Section 5 shows similarities and differences to other modeling approaches. Section 6 finally concludes and gives information about work which still has to be carried out in the future.

[1] http://code.google.com/intl/de-DE/webtoolkit/
[2] http://code.google.com/intl/de-DE/webtoolkit/tools/gwtdesigner/
[3] http://www.uml-lab.com/

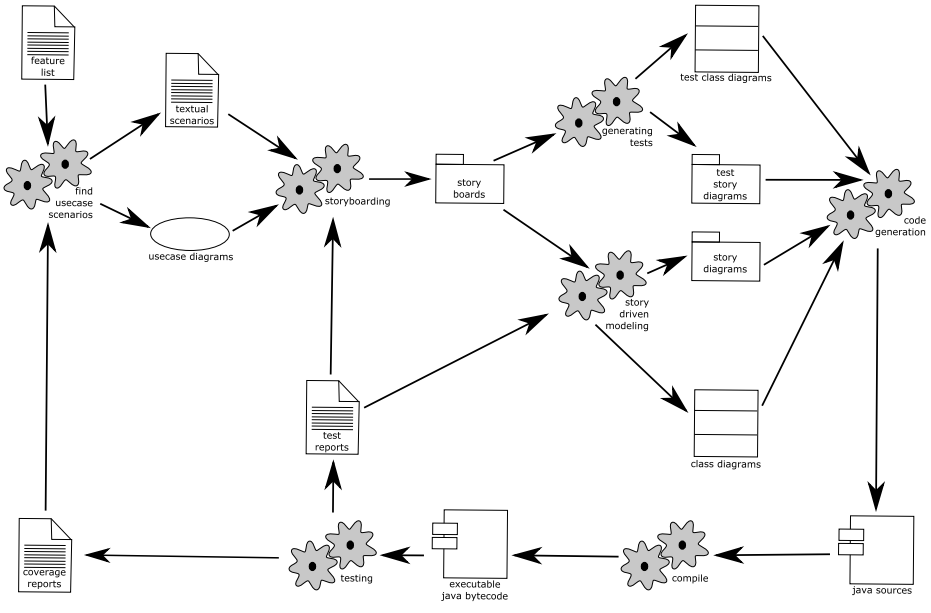


**Fig. 1.** The traditional Fujaba Process used for story-driven development of traditional applications

## 2 Targeted Applications - What We Call Enterprise Web Applications

As stated in section 1 our web application development process ist tailored to a special class of applications. We call these applications Enterprise Web Application and define them as follows:

- **Ajax based applications with excessive Document Object Model restructuring.** We explicitly do not target page based applications with traditional link infrastructure such as HTML links. Enterprse Web Applications will change the "page content" by resturucturing the Document Object Model of the web page. This way we do not have page reloads. This behaviour is also known from Rich Internet Applications.
- **Client side data model and logic.** We try to shift as much of the business logic of our applications to the client. The web browsers Ajax Engine takes care of computations and data model changes. This way, most of the application logic can be shifted to the client. The applications will only contact the server in special cases.
- **Desktop-like user interfaces.** Enterprise Web Applications have interfaces composed of so called widgets. We have controls similar to the ones known from desktop applications. Lists, buttons, dropdown-boxes and menus are used to provide access to the data model and logic.

- **Minimized server side code.** As stated above, we try to limit the server computations wherever possible. Only security critical issues are still computed by the server and persistent storage is carried out using the WebCoobra persistency and data replication framework.
- **Workflow driven.** This last item is no must. However, we state that most applications are developed to provide tool support for some kind of workflow. This is why we have introduced the workflow driven requirement into our application class as well as in our process. Workflow hereby does not mean to be forced to describe the work to be carried out using a Business Process Model. Nevertheless, recurring tasks may also form a workflow in some ways.

As example for what we call Enterprise Web Applications, the GoogleDocs[4] may be reviewed. While this application is also able to have multiple simultaneous users on one document, the technology used for this differs from our approach. However, the requirements to be of class Enterprise Web Application are fulfilled by GoogleDocs. Being similar in many cases, in contrast to traditional software the Enterprise Web Applications are accessible from everywhere in the world without the need to be installed. Being more flexible, this way, they still contain similar features than desktop software and are even more complex to develop due to the distributed nature. The story-driven development of this kind of applications is target to our research and development process.

## 3 Story-Driven Modeling of Enterprise Web Applications

The story-driven modeling approach is taught and researched by the Fujaba community for years now. There also exists a process, defining the steps needed to develop applications this way, the FUjaba Process (FUP). Figure 1 shows the complete process. The research of story-driven modeling of Enterprise Web Applications was based on this existing work, as described above. However, our targeted web applications still differ from the kinds of applications the FUP was targeted to. Nevertheless, FUP can still be used for the server parts and for parts of the data model of Enterprise Web Applications. A description of application development accourding to the FUP can be found in [3]. In the following, we will only give a short introduction on the main story-driven modeling features.

The development process starts by the definition of textual scenarios. One example scenario could be the following: *"Alice and Bob are playing Ludo. It is Alice's turn. She has rolled the dice which shows 2. Alice takes her red piece and moves it forward two fields."* The textual scenarios will then be translated to so called storyboards. Figure 2 shows the storyboard resulting from the example scenario shown above. These storyboards are used for automatic test generation. One Test is generated from every storyboard, testing the described scenario. Additionally, storyboards serve as starting point for the application development. As can be seen from Figure 2 the method `move(2)` is called on the `piece` object. This method can be implemented graphically using storydiagrams inside

[4] http://docs.google.com

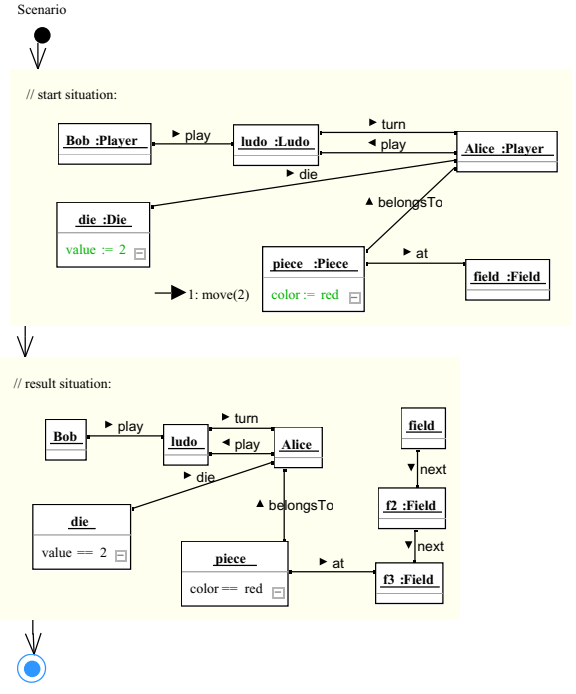


**Fig. 2.** Storyboard describing the example textual scenario of Alice and Bob playing ludo

the Fujaba Toolsuite. This way, the logic and data model changes are handled. The `move()` method will change the data model of the application by placing the `piece` object on a different `Field`. The diagram showing an example implementation of the `move()` method is shown in Figure 3. The Fujaba story-driven modeling techniques and capabilities are described in detail in [14]. Using the Fujaba Toolsuite and process, we are able to create classdiagrams for the data model as well as application logic using storydiagrams. All of these diagrams are used as input for the code generation process. This code generation results in Java source code implementing data model and application logic. The source code can be compiled using standard Java compilers and afterwards be executed within the Java Virtual Machine. The development of highly complex systems is possible this way. As an example: The code generation engine of Fujaba is bootstrapped and was developed using story-driven modeling approaches, itself.

The Fujaba development process described above in context of Enterprise Web Applications sufficient only for the server part and data model. There are some major issues missing in FUP concerning these kind of applications. The modeling of distributed systems is not clearly defined. These systems contain client and server part as well as communication between these parts. Additionally, the data model, which is created using the FUP techniques has to be bound to the client

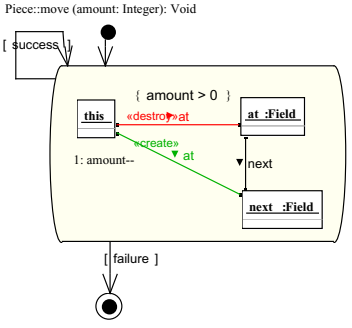


**Fig. 3.** Activity Diagram modeling the logic of the `move()` method

side user interface in some way. First steps towards the modeling of server calls and data bindings were introduced with the Fujaba *Action Charts* [5]. However, we have to provide modeling capabilities for the issues missing in this approach. The process for Enterprise Web Applications had to provide information on how to model server calls, do the databinding and create controller logic for the user interface. The implementation of graphical user interfaces is a painstaking task. This holds for hand coding as well as for modeling UIs with storydiagrams. However, we need support for visual user interface design in our process. Fortunately, the publication of Googles GWT-Designer[5] in 2010 opened a way for visual user interface definition. While the definition of UIs using the GWT-Designer makes life easier, the output of the graphical editor is Java source code. Since we intend to have a completely model-driven development process of Enterprise Web Applications we do not want to switch back to source code. We thus had to incorporate some kind of automatic mapping at this point. Following our process and using the associated toolchain we are able to generate source code, cross-compile it with the Google Web Toolkit and run a completely modeled web application. As a result of our research on the development of Enterprise Web Applications this paper presents our story-driven modeling approach defined in the Web Fujaba Process (WFUP).

### 3.1 Running Example

While the whole approach presented in this paper is still conceptual in some points, we needed an example application to test the whole process. This application had to be complex, support multiple users and have different user interfaces (views) for different kinds of tasks. Since Enterprise Web Applications will mostly be used to support some kind of workflow, as stated in 2 there also had to be a workflow with sufficient complexity to be solved by the example application. We choose a management workflow for the creation of animated computer films. The Enterprise Web Application modeled to support the whole

[5] http://code.google.com/intl/de-DE/webtoolkit/tools/gwtdesigner/index.html

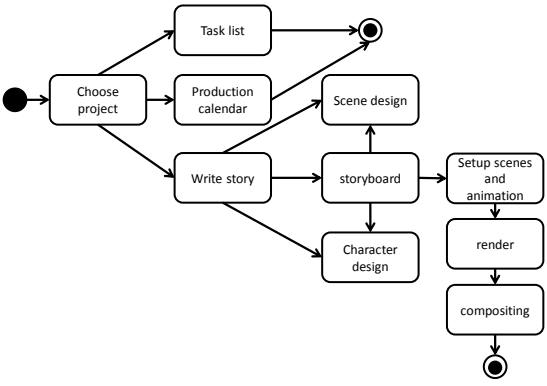


**Fig. 4.** Workflow of the example application to manage the creation of animated films

management of film productions like this will need to support the whole workflow shown in figure 4.

The application will need to manage different film projects from which the user can choose when starting the application. After this, information for the choosen project will be shown. Every step of the workflow shown will have its own user interface (view). The views will consist of similar items as user interfaces known from desktop applications, e.g. a calender with information about the project status, giving the project manager some kind of timely overview. Another view, the write story view (shown in Figure 5), will consist of a web editor, which will enable the user to define scenes, dialogues and the action taking place within the scene. This is done textually. Not every task of the workflow will be done within the webbrowser, 3D modeling, design and animation will as well be done with specified desktop applications as rendering and compositing processes. The web application will only show the current status of these action points. The example application serves as basis for the development of tool support for the proposed process. Therefore, we will develop the application according to this process. However, some parts of the application or process may be changed during development and further research. The process introduced in section 4 is the first version and will need further refinement. Even the workflow of the application might be changed, rearranged or further enriched. Maybe it might be necessary to define subworkflows for more complex action points.

## 4 The Web Fujaba Process (WFUP)

As a result to the question raised in section 1: “Is it possible to develop web applications without the need to write any sourcecode, too?” we propose the Web Fujaba Process. Combining the experience from traditional story-driven modeling and the research work introduced in section 3 we propose this process as guidance for the completely story-driven modeling of Enterprose Web Applications. The Web Fujaba Process hereby is based on the traditional Fujaba

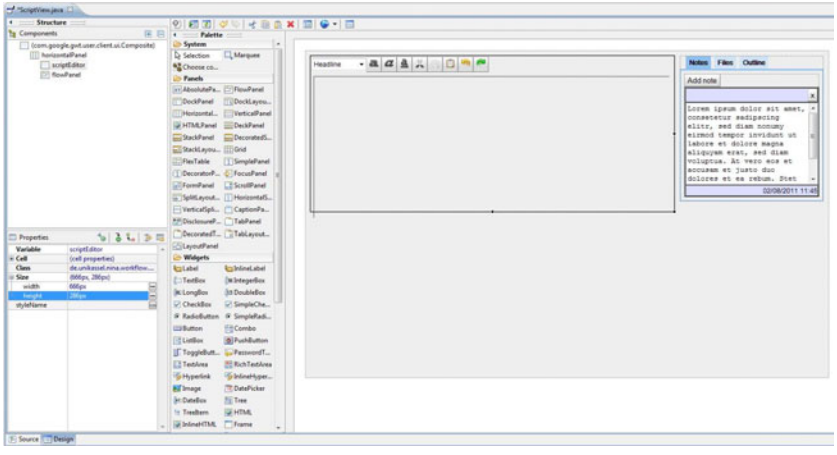

**Fig. 5.** Write story view from example application within GWT-Designer view in Eclipse

Process shown in figure 1. The Fujaba Process was further enriched with the description of user interface design and the possibilities to model UI controllers as well as server calls for distributed applications. However, the original process is still part of the development. The server parts of applications will be developed according to the original process. Figure 6 shows an overwiew of the Web Fujaba Process (WFUP).

The process starts by describing the intened application textually, as it was done in the old process. Taking the description as input, usecase scenarios are created and documented as well textually as in usecase diagrams. This step should be carried out together with the potential users or the customer. Since Enterprise Web Applications will be divided into a client- and a server part the process is divided at this point, too. The server part takes its textual and usecase scenarios and is further developed according to the FUP. This also is done for the application data model. This model can be designed the traditional way, resulting in class diagrams which can in turn be used inside the WFUP to do the data binding and controller parts. Since the applications we intend to create with the WFUP will deploy the data model both on server side and on client side, it is extremely necessary here, to integrate the model into both development steps. For the server part, the application data model can be used as in the development of traditional applications with the FUP. However, the controller structure will be able to do changes on the data model directly in the client, without the need to make server calls. This results in the need of having the class diagrams of the data model in the WFUP process, too. Having special annotations and interfaces in the application data model will enable the use of the WebCoobra Framework [1] for the data model. This gives us support for automatic replication of data between the server and multiple clients, enabling the system to keep consistency. These annotations and interfacing can easily be

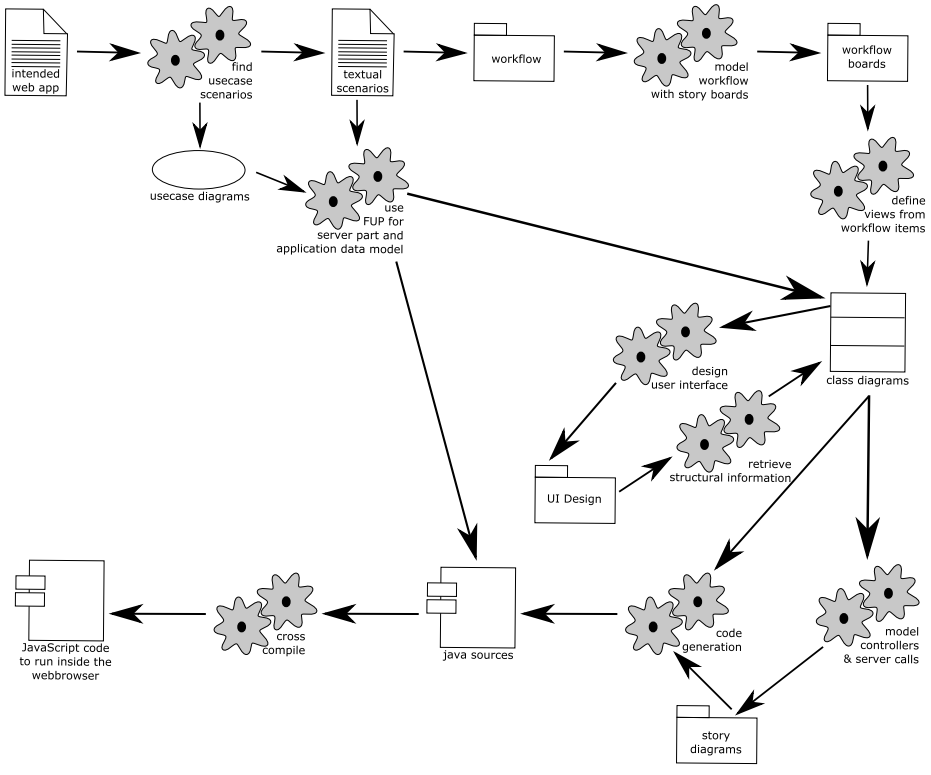


**Fig. 6.** The Web Fujaba Process, a proposal for story-driven development of Enterprse Web Applications with the Google Web Toolkit and Fujaba

done during the FUP for the server part and data model and do not have to be taken into account while developing the client side with the WFUP. The clients side code can handle every object of the data model directly, this way, which eases the modeling of controllers for the user interface. The client part of the application will be developed using the additional steps introduced in this paper. Enterprise Web Applications mostly will follow some kind of workflow. This is due to the case that applications will mostly be designed to carry out some kind of work. In our running example introduced in section 3.1 the application to be developed is designed to enable the management of animated film productions. Thus, there is a workflow driving the application - the steps to be carried out to manage an animated film production in our case. This is true to the major part of applications. Hence, we try to extract a workflow from the textual scenarios. This workflow is modeled using Fujaba and will result in so called workflow boards. For every workflow step there will be one graphical user interface instance - one view. The view for the workflow step should be designed to support the user in performing the task associated with the workflow step. To enable this, we define one view component for every workflow step. The view components are simply UI

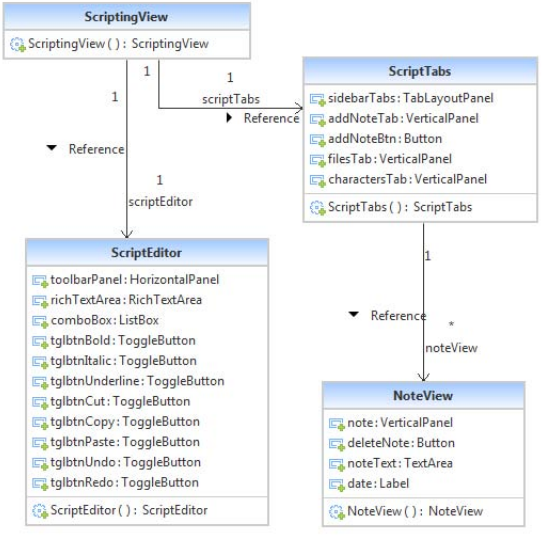


**Fig. 7.** Reverse engineered classdiagram of script view user interface. This reverse engineering step is carried out by UML Lab, using the Java source code generated from the Google GWT-Designer.

classes inside a class diagramm. To link between the view class and the associated workflow step, we use a new kind of diagram, the UI board. The next step would be to further design the user interface. This will be done using the Google GWT-Designer. In a graphical way the user interface will be created and saved. As stated in 3 the GWT-Designer creates Java source code as output. We therefore retrieve back the structural information of the created classes to be able to use it within the remainder of the process. Figure 7 shows the classdiagram of the example view from Figure 5 after retrieving back the structural information from the GWT-Designer. The resulting class diagram has referenced classes to all the used user interface components as well as all the attributes set. Additionally, the structure of the view is represented by links between the different user interface classes in the diagram. After this step is finished, the structure of the view can be combined with the data model for the application, enabling the controller and application logic to perform model changes. The simplified version of the data model for the example application is shown in Figure 8. The logic of the controllers as well as the databinding can be modeled with Fujaba storydiagrams using the information of both of the classdiagrams shown above. The controllers in turn will result in a third classdiagram containing the controller structure. Every controller class will have methods which are automatically called by the associated user interface components when actions occur. The methods of these controller classes can be modeled with story driven techniques in Fujaba. After

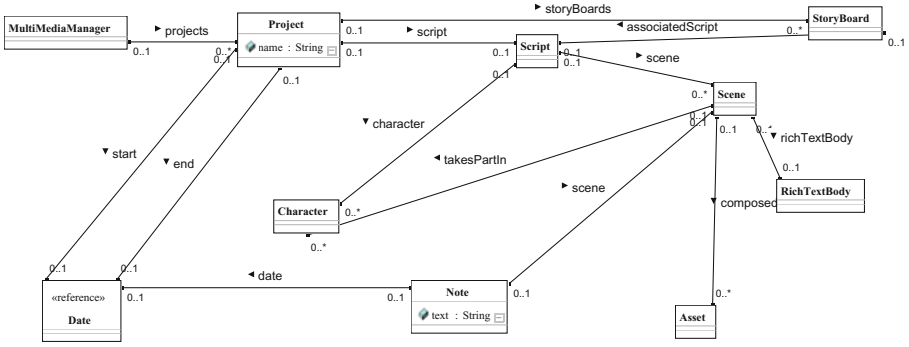


**Fig. 8.** Simplified data model for the example application in Fujaba notation

the modeling process is done, the runnable application will be created using code generation mechanisms.

The process described above enables the complete modeling of web applications including user interface components and client as well as server side logic. To support the user in carrying out this process, we started to implement tool support for this process. The suggested toolchain is further described in the following section.

### 4.1 WFUP Toolchain - Adding Tool Support to Our Process

The Fujaba Toolsuite already supports class and storydiagrams. However, the user interface designing process using Google GWT-Designer produces source code. We therefore need some kind of mapping from this source code to Fujaba's model here. We choose to retrieve back the structural informations of the designed user interfaces into Fujaba, here. This way, it would be possible to run further process steps with this structures. We found the tool support for this step in UMLLab. This tool is able to reverse engineer source code into structural model information and provide real-time synchronisation between model and code. The structural information gained from UML Labs reverse features enables us to close the gap between source code and model at this point. Still, the structural model gained through UML Lab does not help in the Fujaba development process because of incompatible meta models of both tools. Additionally, it is not possible to model the logical parts of web applications with story-driven modeling using UML Lab, yet. To overcome this problem, a real-time adapter between UML Lab and Fujaba has been created [6]. Using this adapter the structural information can be duplicated into Fujaba class diagrams and afterwards be used for the story-driven modeling of client side logic, controllers and databinding. The adapter also takes care of consistency between class diagrams residing within Fujaba and those within UMLLab. This means, we are able to apply changes to any of those diagrams later. To enable more comfortable access to the controllers, we have started to add right-click menus to the GWT-Designer. This way, we can add controllers to our user interface in a

graphical way and only need to implement the logic of the controller afterwards. Having defined the whole application, the process takes the resulting diagrams; class diagrams of the view, story diagrams for the controllers and for server calls and generates code for all of these diagrams. Different code generation strategies will be used here, depending on the type and semantics of the diagram to be generated. This code generation step will result in Java source code that can be tested and debugged with the Google Plugins and which can be cross-compiled with the GWT cross compiler to gain the JavaScript code that is run inside the webbrowser and which can be deployed onto an application server. The tool support for the Web Fujaba Process is not yet completely implemented. However, the design of user interfaces, retrieval of structural information and design of user interface controllers and server calls can already be done using current versions of GWT-Designer, UML Lab and Fujaba.

## 5 Related Work

The approach and process proposed in this paper is not the only research work in the domain of model-driven development of web applications. However, we face on the development of a special class of applications: Enterprise Web Applications. Additionally, we want to be able to do this with the tool support we are used for the implementation of traditional applications. Thus, we try to broaden our modeling and graph transformation expertise to the domain of web application development. However, we will try to sum up some of the similarities and differences to other research work in this sector in the remainder of this section:

[7] shows the use of patterns for the development of Rich Internet Applications. The authors insist to use patterns on a higher abstraction layer, meaning in the modeling stage. The approach uses models of these patterns that are modeled once and then reused by model-to-model transformations or manually. Similar to our approach of Action charts [5] the authors introduce UML state diagrams for the modeling of user interface actions. The reuse of some patterns modeled to execute common user interface operations will be further invesitigated and might be included into our process in the future.

The Orchestration Model which is introduced in [10] in combination with the "Model-Driven Development Process for RIAs" follows similar steps than the process introduced here. However, there is a lot of model transformation between platform independent and platform specific models needed, following this approach. These meta layers often get confused as well from developers as customers. We have tried to hide these meta-information from the customer using our story-driven approaches in the past. Tests with students showed that application development following the story approach often is simpler for the customers and developers to understand and in additon there is always a document which can easily be understood by both. The simple notation of storyboards and usecase diagrams in some points enabled the customer to design parts of the development process on his own.

[12] contains activity diagrams for the generation of user interfaces as well as WYSIWYG user interface design. These diagrams are similar to our workflow

boards. However, we will use the information flow of workflow boards to generate the exchange of one view or parts of it by another. The user interface design will, like in [12] be carried out with graphical design. However, the approach introduced in [12] is designed for Enterprise Applications using Eclipse and SWT rather than web applications.

The UWE4JSF method [8] again uses similar approaches than the WFUP introduced above. However, it faces on JSF[6] applications rather than real Ajax applications. The navigation structure shown in this paper can be seen similar to our workflow boards, again. Nevertheless, transitions within our workflow boards will not trigger page changes. It will rather result in the Document Object Model Tree to be changed, meaning that part of the views or even complete views will be exchanged without the page to be reloaded.

[4] shows ways for multi-level testing of model driven web applications. This point will be start for future research of WFUP. While the traditional FUP enables testing and even automatic generation of test code, the WFUP currentls lacks this for the client side of applications. We will have to find ways to (semi)automatically incorporate tests into the WFUP in the future.

Another interesting idea is the model-driven way of importing user interface mockups into the real development process described in [11]. The implementation of this idea into our intended toolchain would really be nice to have. However, since the GWT-Designer is not yet open-source there is currently no plan to implement this into our process.

WebML is widely used in the domain of model driven web engineering. The approach and process presented on [13] are used to define applications which do not fit into our targeted application class as defined in section 2. While we target on Enterprise Web Applications without page reload and with client side application state, WebML is mainly targeted on traditional page based web applications. Thus, the process defined by WebML is not useful for the story-driven development of applications we want to create. We therefore defined our own process, which is exactly tailored to the needs of Enterprise Web Applications.

As far as we know, the completely story-driven modeling of web applications was not yet done in the community of web engineering. We tried to introduce the process we are used to and adopt it to the special needs of our targeted application class. Getting input from additional research in the are of web engineering will hopefully further enrich this process in the future and make it less experimental.

## 6 Conclusion and Future Work

This paper showed the research steps and methods carried out to design a completely story-driven approach for the development of Enterprise Web Applications. The Web Fujaba Process (WFUP) was introduced as our proposal for the development of these applications. The predecessor, the Fujaba Process, used for the development of traditional desktop applications was introduced. The changes

[6] Java Server Faces - http://java.sun.com/javaee/javaserverfaces

and enhancements made to extend this process for the development of Enterprise Web Applications including user interface design were shown. The work to be carried out to completely implement tool support and user friendly usage of the proposed tool chain is still going on. Since we intend to have a development process that resides completely within the model, without switching to source code writing, we still have some minor problems to face. At the moment there is still the need to combine different tools to develop applications. GWT-Designer for the user interface specification, UML Lab for the sturctural information retrieval, Fujaba for the main development of the rest of the application. While all of these applications can be combined at this point, there is still the need for better tool integration. The diagrams for databinding and controllers of user interfaces, as well as the ones for server calls will need some rework. Some new diagrams might be introduced for these purposes, in case we figure out that the traditional story diagrams will not suffice for our proposal. Additionally, there will have to be some wizards for the generation of complete web applications as well as for the creation of different kinds of diagrams. Having new diagrams will also mean to have to update code generation to fulfill the needs of web applications. There will be different semantics of storydiagrams, so we will have some kind of context-sensitive code generation here, depending on the purpose of the diagram. While there has to be done a lot of work with the distributed parts of the web application, the server parts can be designed and generated following the current standard FUP process. This means, there is automatic generation of test for the server part of the application. Hence, we will try to enable test generation for client parts as well. Since this is current research work, we do not yet know which ways of testing will be used within the web application process of Fujaba. There is still a lot of work to be carried out to enable complete story-driven development of web applications and have all the different tools and parts integrated well into each other. Nevertheless, we still want to stay with the model-centric and story-driven approach. This way, we hope to bring the common advantages of model-driven development to the web domain. The diagrams we propose to use might, in addition, often be better to read and understand by customers than this is the case with source code.

## References

1. Aschenbrenner, N., Dreyer, J., Hahn, M., Jubeh, R., Schneider, C., Zündorf, A.: Building Distributed Web Applications based on Model Versioning with CoObRA: an Experience Report. In: Proc. 2009 Intl. Workshop on Comparison and Versioning of Software Models, pp. 19–24. ACM (May 2009)
2. Diethelm, I., Geiger, L., Zündorf, A.: Systematic Story Driven Modeling. Technical Report (February 2004)
3. Diethelm, I., Geiger, L., Zündorf, A.: Systematic Story Driven Modeling, a case study. Edinburgh, Scottland (May 2004)
4. Fraternali, P., Tisi, M.: Multi-Level Tests for Model Driven Web Applications. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 158–172. Springer, Heidelberg (2010)

5. Geiger, N., George, T., Hahn, M., Jubeh, R., Zündorf, A.: Using actions charts for reactive web application modelling (2010)
6. Koch, A.: Echtzeit synchronisierung von uml-modellen unterschiedlicher technischer basis am beispiel von uml lab und fujaba. Master's thesis, Kassel University, Fachbereich 16, Fachgebiet Software Engineering, Wilhelmshöher Allee 73, 34121 Kassel (September 2010)
7. Koch, N., Pigerl, M., Zhang, G., Morozova, T.: Patterns for the Model-Based Development of RIAs. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 283–291. Springer, Heidelberg (2009)
8. Kroiss, C., Koch, N., Knapp, A.: UWE4JSF: A Model-Driven Generation Approach for Web Applications. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 493–496. Springer, Heidelberg (2009)
9. Nickel, U., Niere, J., Zündorf, A.: The Fujaba Environment, Limmerick, Ireland, pp. 742–745. ACM press (June 2000)
10. Pérez, S., Díaz, O., Meliá, S., Gómez, J.: Facing interaction-rich rias: The orchestration model. In: Schwabe, D., Curbera, F., Dantzig, P. (eds.) ICWE, pp. 24–37. IEEE (2008)
11. Rivero, J.M., Rossi, G., Grigera, J., Burella, J., Luna, E.R., Gordillo, S.E.: From Mockups to User Interface Models: An Extensible Model Driven Approach. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 13–24. Springer, Heidelberg (2010)
12. Schramm, A., Preußner, A., Heinrich, M., Vogel, L.: Rapid UI Development for Enterprise Applications: Combining Manual and Model-Driven Techniques. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010, Part I. LNCS, vol. 6394, pp. 271–285. Springer, Heidelberg (2010)
13. The Web Modeling Language (2011), `http://www.webml.org`
14. Zündorf, A.: Rigorous object oriented software development. Habilitation Thesis, University of Paderborn (2001)