Object-oriented Analysis and Design Approach for the Requirements Engineering

A. Zeaaraoui, Z. Bougroun, M.G Belkasmi, T. Bouchentouf Belkasmi
Laboratory of Applied Mathematics
Signal Processing and Computer Science
Department of Computer Science, ESTO
Oujda, Morocco
adilzeaaraoui@yahoo.fr, tbouchentouf@gmail.com

ABSTRACT: *In the software development process, developers feel the gap when moving from requirement engineering phase (using scenario-based approach) to construction phase; this is due to that models result in RE cannot be easily mapped to models in construction phase. This paper discusses this problem, and offers a new approach that handles this issue so that developers feel no break during all software development activities from RE to coding and testing.*

Keywords: Requirements Engineering, Object-Oriented Applications

Received: 12 July 2012, Revised 10 September 2012, Accepted 15 September 2012

1. Introduction

Requirements Engineering (RE) is the process in which different types of methods and technics can be used to obtain the requirements of a system and its properties.

'*Requirements engineering process*' has the most dominant impact on the capabilities of the resulting product, because an incomplete or poor RE affects the quality of the final software[2], and errors are more expensive to fix later in project lifecycles[12; 13]. This is why many approaches have been developed and many efforts have been made to enhance this process (Barry Boehm investigation). Requirements engineering process is composed by seven main activities: project creation, elicitation, interpretation and structuring, negotiation [4]. Verification & validation, change management [3].and tracing [11]. After RE phase, when developers move next to software construction phase (domain model, class diagram and so on), they feel the gap between this two phases; this gap is presented as a non-possibility to map from "*scenario-based RE "* [1]. models to object models as shown in figure 1.

This paper discusses this problem, and presents another approach which is: object-oriented analysis and design approach for requirement engineering.

2. Scenario-Based Approach for Requirements Engineering

Requirements engineering is all the technics, skills and methods, automated or not, used to collect functional and non-**functional** requirements of users allowing them to achieve their objectives that must be possessed by a system.

The well-known mean of exchange between users and developers (or designers) used to collect requirements, is scenario. There are

many forms to express scenarios, but all of them present only functional and behavioral aspect of requirements. Before presenting weaknesses of scenario-based approach in object-oriented software development, we title its forms as follows: *User stories structure, use case model, sequence diagram* [5]*., use case map* [6, 8]*. story-board.*

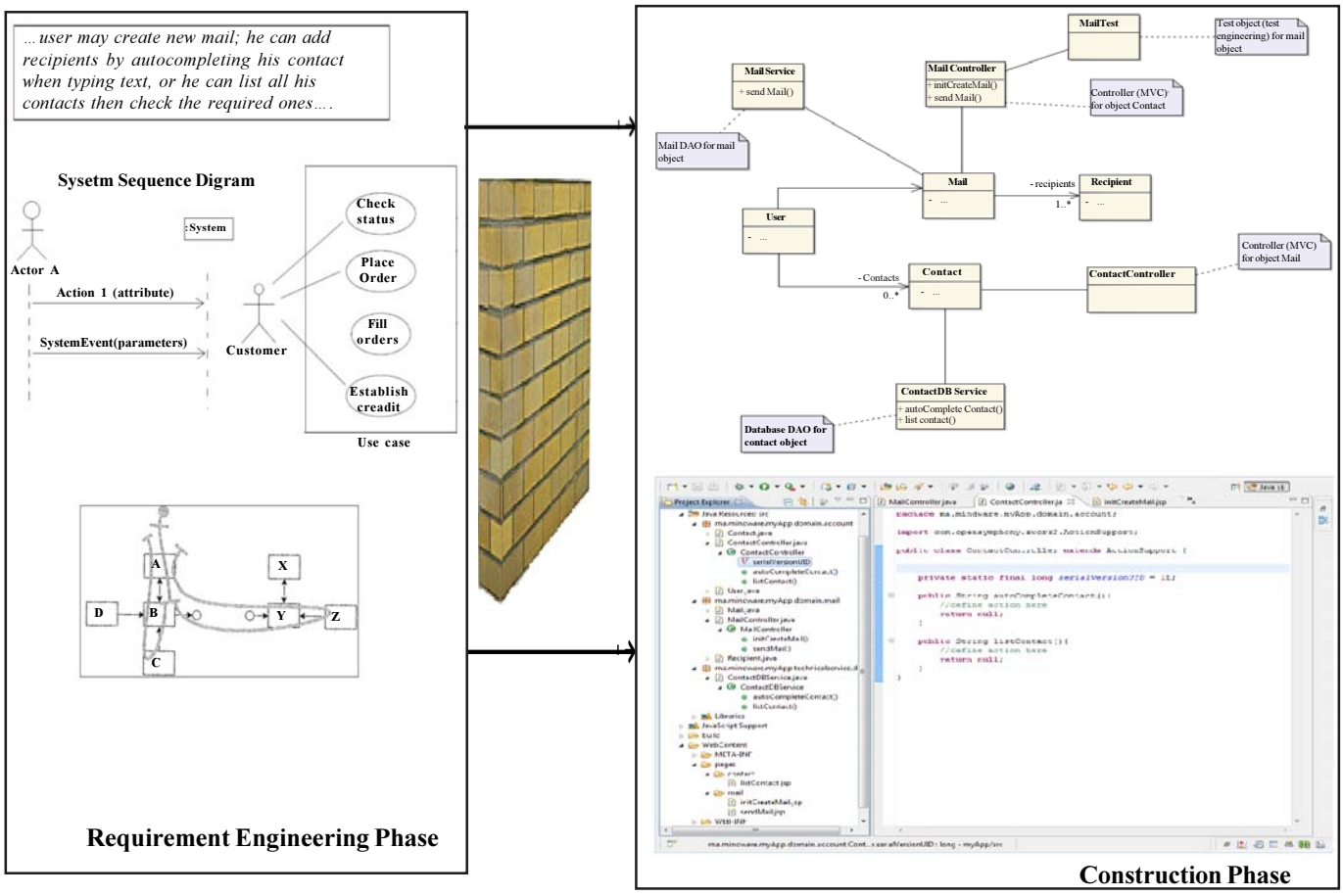

Figure 1. The break flow when moving from RE phase to the construction phase in software development

2.1 Weaknesses of scenario-based approach

As we noticed about scenario-based approach, we have only the functional nature of the system; for object-oriented software in which this research is interested, it is a big issue because:

• **Objects and functions do not map to each other:** the architecture of functionally decomposed system is significantly different from the architecture of an object-oriented system; from UML point of view, it is impossible to map directly from use case model, sequence diagram, system sequence diagram…etc. to domain model or object model and vice versa. Also using use cases within the context of object-oriented development - so called "*differing localization strategies*" - will result in the introduction of significant errors [9].

• **Charge estimation problem:** the effort to develop a use case (scenario) can vary from a day to a year; if you go with the one-paragraph approach to use cases, there is enormous room for ambiguity so one cannot have clear path view of charges. Because use cases (scenarios) are so general and considered a high-level user-interface design, or a high-level design of operating procedures, so that they provide almost no guidance.

• **Do not enable organizing complexity of software system:** object-oriented decomposition directly addresses the inherent complexity of software by helping us make intelligent decisions regarding the separation of concerns in a large state space [10].

To resume, scenario-based approach showed weaknesses in software development process, this does not say they are not valuable, but they give functional presentation of our system, and readable way, "*what do*", of our system for the customer.

3. Object-Oriented Analysis and Design Approach for Requirements Engineering

3.1 Introduction

Even the come of processes, methodologies…etc., many stakeholders (companies, students, developers…) are facing the problem to move from requirements engineering phase to construction step in a way there is no break in development activities flow for object-oriented software; in addition to that, none of them showed an automated practice for software industry or a practice that developers may apply for new projects, maintenance…etc. [14].

The important activity and most early step in object-oriented software construction phase is the domain model; this model is converted to PIM (Platform Independent Model) as class diagram that will be transformed to PSM (Platform Specific Model), then generate code (classes) to begin actions implementation. This gives a light that domain model is valuable and fired in mind the idea that we must think object, we must concentrate on objects and consider them as the core and as the backbone when analyzing and designing.

Object-Oriented Analysis and Design (Grady Booch approach) is convenient method to build object models from requirements; but this method showed weaknesses in a way it focuses only on object design and neglect capturing what a system must do. Our idea rose from here; is to use OOA/D method and dig to:

- Have a method to capture and identify automatically a "*what do*" the application using check list of actions.
- Make a practice to automate some software artefacts like patterns: mvc, dao…etc.

3.2 How OOA/D approach for RE works

The main objective of this paper is to facilitate, automate and provide a practice that any developer may use in software development. The idea of this approach is simple; it is based on object-oriented analysis and design approach; during object analysis, when an object is identified we define to which category belongs and we check which actions, from the predefined ones, must fulfill this object by understanding user goals. Below a table presenting examples of object categories and their actions:

Object Category	Description	Actions
Ordinary	**Ordinary objects are objects that share ordinary actions, and have a common characteristic functionally thinking. Examples: User, Car, Right, Role, Product…**	☐ initCreate ☐ create ☐ initUpdate ☐ update ☐ get ☐ list ☐ search … delete
Mail	**Mail objects are object that are different from ordinary objects, and they offer actions related to mail. Ordinary objects and mail objects share some actions.** **Examples: Mail, Message**	☐ initCreate (or initSend) ☐ create (or send) ☐ initUpdate ☐ update ☐ get ☐ list ☐ search ☐ initReply ☐ reply … acknowledge

The database of categories and their actions may grow by experience and feedback. Actions from the table are self-explanatory, for example the action "*initCreate*" means initialize the creation of the object in question and prepare the view (initialize data) for the creation. To clarify deeply this approach and see how all of this is transformed to code, let's project it on a web based application.

4. OOA/D Approach for Requirements Engineering in web-based application

In this section we will see a web-based application developed using this approach; the application is called "*MindMail* " (http://

www.mindware.ma/product/getProduct?productID =1&MindMail-E-mail-&-Collaboration) which is an open source web-based mail & collaboration tool developed by MindWare Company (http://mindware.ma).

To present how this approach is used in this application, we take a short part of user-story and detail all steps followed from OOA/D RE to coding. During design some quality and design artifacts will be undertaken, like MVC (Model View Controller), DAOs (Data Access Objects) and so on , but how this can be related to our approach? Good, here is the idea: when capturing an object, we create its controller holding its actions (mvc), create one or many of its DAOs, its test class…etc. Figure 2 illustrates the design of this approach in a class diagram.

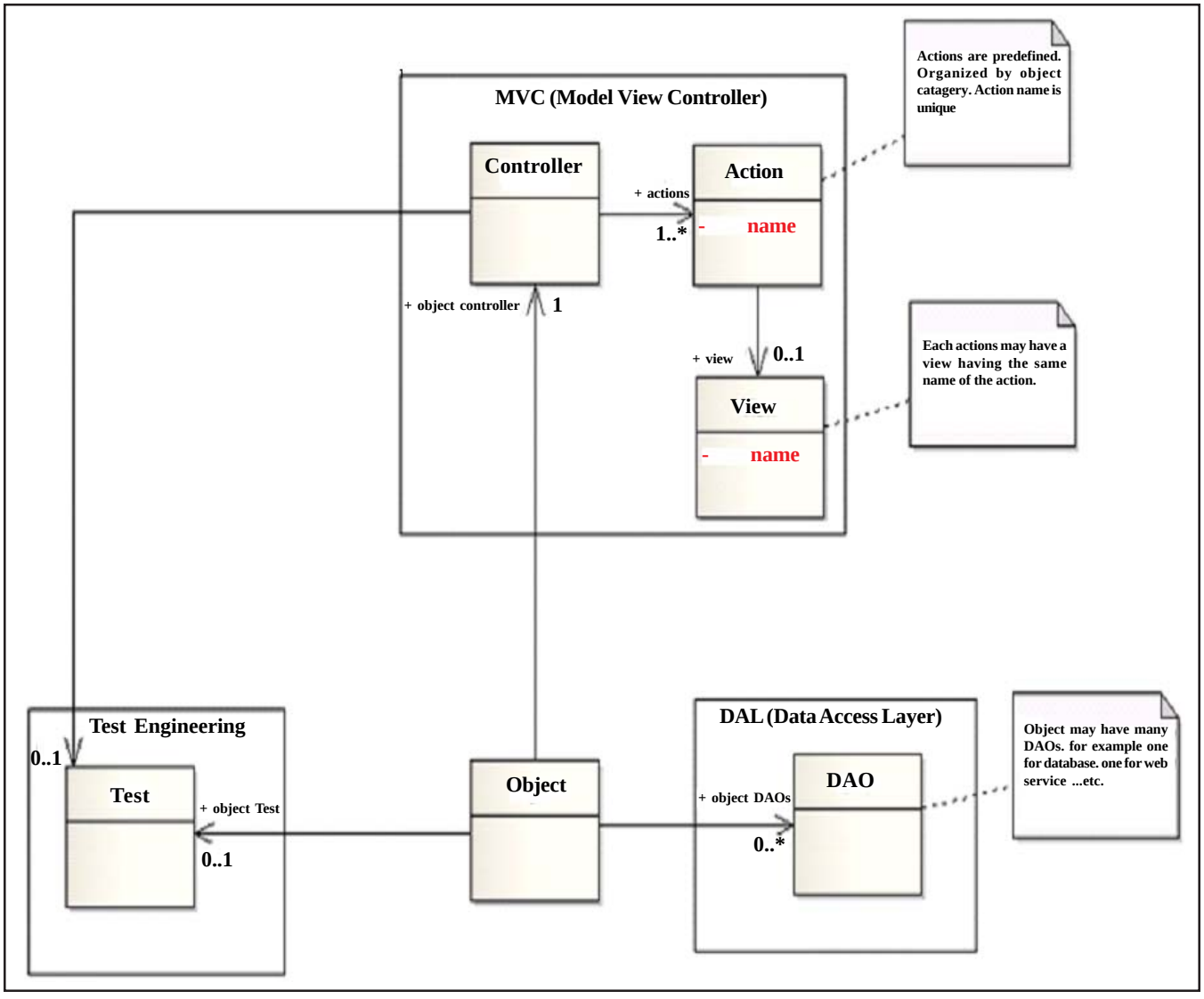

Figure 2. Class diagram showing how the main element "*object*" is related to other design artifacts

To clarify how to develop any kind of web-based application using this approach, we move to show steps followed giving examples for each one.

• **Step 1:** form user stories In this step a set of interviews, workshops…etc. must be done to gather and elicit all functional requirements [3]. then write the gathered information in a well formed ordered text using a developer style (user stories). For our example we take a part from the user stories about creating new mail:

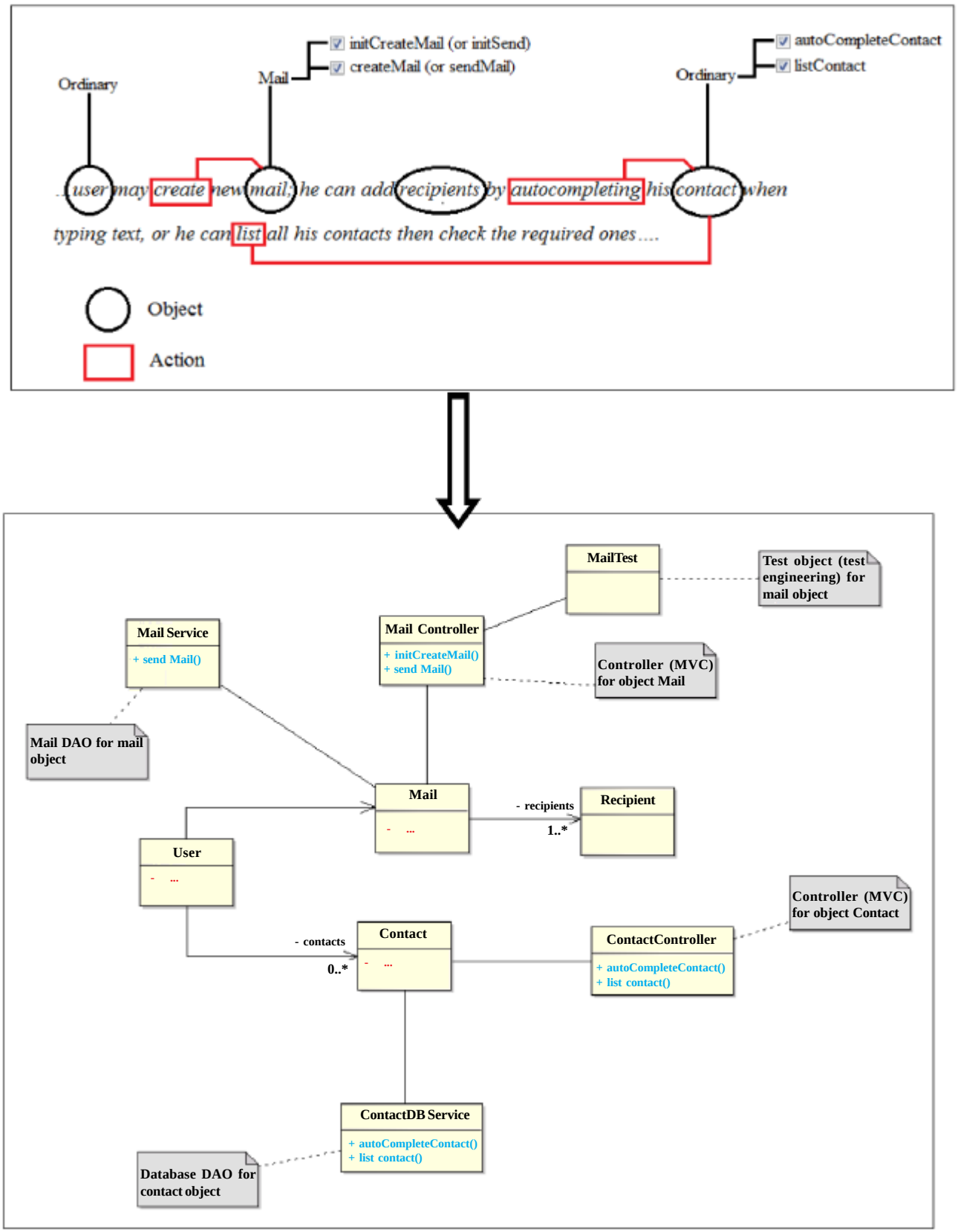

Figure 3. Illustration of how class diagram is constructed from user stories using OOA/D Approach for RE

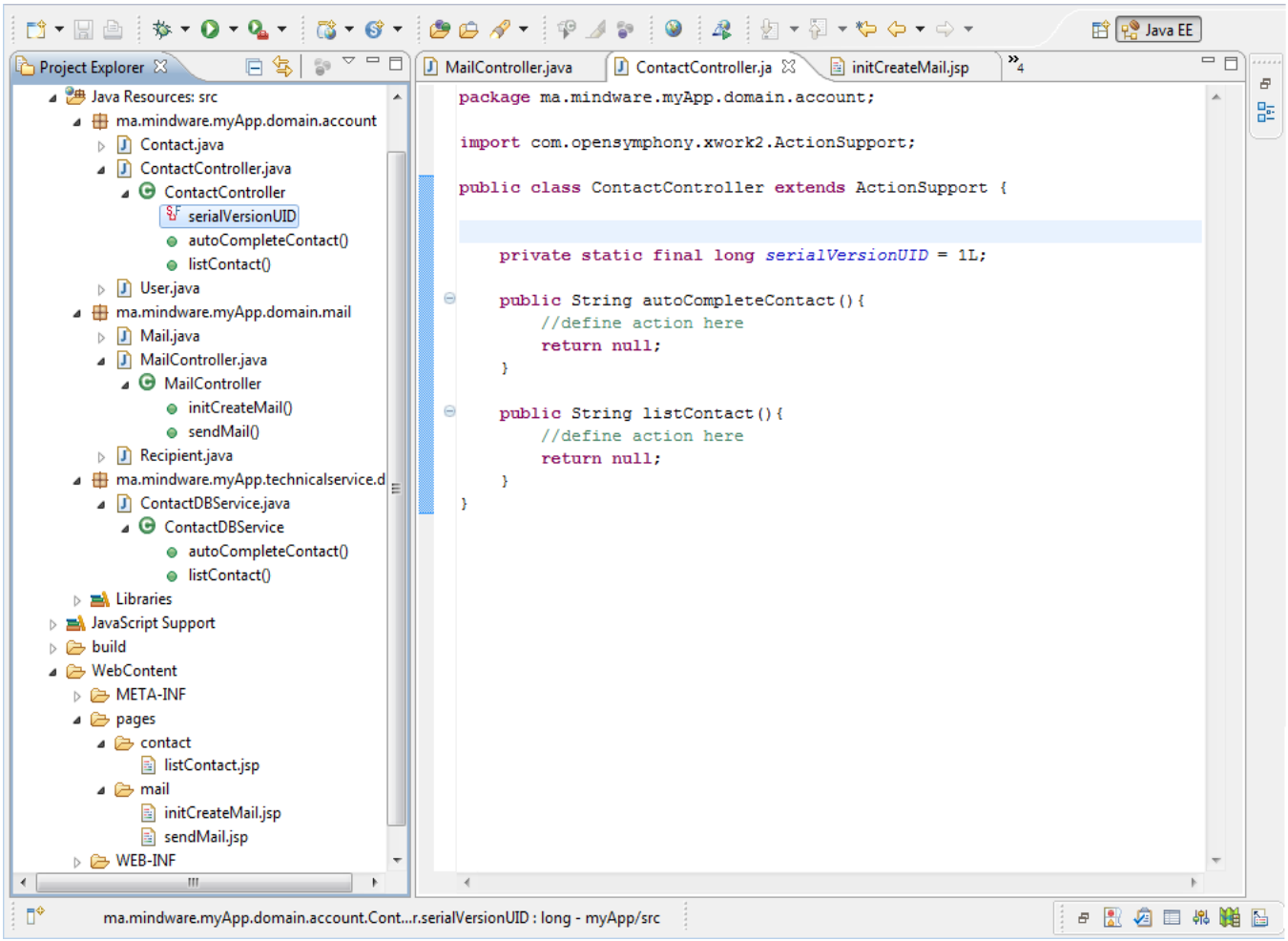

Figure 5. Screen shot showing the organization of the generated code from using OOA/D for RE

> *…user may create new mail; he can add recipients by autocompleting his contact when typing text, or he can list all his contacts then check the required ones….*

• **Step 2:** OOA/D RE

In this step, first we begin by highlighting entities, then identifying objects one by one and define to which category belongs each one, after that, as we said we must think "*object* ", we capture from the text what must that object offer, what actions must be ran on that object to answer user's goals, then check the appropriate actions from the predefined ones. While doing this, we project and model the gathered elements (object, action, connection, interaction…) on a class diagram by creating objects, their controllers with actions, their DAOs and so on. Let us see the practice:

• **Step 3: generating code**

Generating code is an implementation phase of our software; this step is added just to give an idea how the code will be organized. Before generating code from the class diagram, we must detail the diagram (methods, data types, attribute and so on.) and specify what the targeted technology is (java, php, .net and so on.); after that, we do the generation using a CASE (Computer-Aided Software Engineering) tool supporting this feature. In our example, as we noticed we have given just the skeleton of the main elements so that one can understands the approach and its steps. Here is a screen shot taken from eclipse showing the organization of the application

and little code generated using JEE with struts 2:

After this step, developers begin building objects actions, then making unit test for each one.

5. Conclusion

Requirements engineering deserves a stronger attention from practice; many approaches were developed to model those requirements, but all of them showed weaknesses in software development process when moving to the construction phase. In this paper we presented methods used to structure and model RE and showed the break they present when moving to construction phase. In the last section we presented our approach, named OOA/D for RE, that fixes this problem and we finished by giving a detailed example using our approach in a web-based application.

6. Future works

While working on this approach, many ideas come to mind to enhance it. Bellow some of them:

- Prototyping a framework based on this approach that details all industry line for software manufactory.
- As we noticed, this approach does not provide a document or a model which can be communicated or delivered to the customer; this must be handled.

References

[1] Salinesi C. Authoring Use Cases. (2004). *In*: Alexander I.F., Maiden N. Coord. Scenarios, Stories,Use Cases: Through the Systems Development Life-Cycle. Wiley & Sons, Ltd, 544 p. ISBN 0-470-86194-0.

[2] Pfleeger, S. L. (1998). Software Engineering – Theory and Practice, Prentice Hall.

[3] Nuseibeh, B., Easterbrook, S. (2000). Requirements Engineering: A Roadmap, The Future of Software Engineering, Anthony Finkelstein (Ed.), ACM Press.

[4] Kotonya, G., Sommerville, I. (1998). Requirements Engineering – Processes and Techniques, John Wiley & Sons.

[5] Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition. Addison Wesley Professional, October, chp. 10, ISBN 0-13-148906-2.

[6] URN - Use Case Map Notation (UCM). (2003). ITU-T Draft Recommendation Z.152. Geneva, Switzerland, Sep. accessed April 2007: http://www.UseCaseMaps.org/urn.

[7] User Requirements Notation (URN) – Language Re-quirements and Framework, ITU-T Recommendation Z.150. Geneva, Switzerland, February, acc. April : http://www.itu.int/ITU-T/publications/recs.html.

[8] Mussbacher, G. (2007). Evolving Use Case Maps as a Scenario and Workflow Description Language. *In*: 10th Workshop on Requirements Engineering (WER'07), Toronto, Canada (May).

[9] Edward V. Berard. (1998). Be Careful With Use Cases, the Object Agency, Inc, (http://www.cs.unc.edu/~stotts/204/usecases/careful.html).

[10] Grady Booch. (1998). Object Oriented Analysis and Design with Applications, second edition, Addison Wesley Longman, Inc., p.16-20, ISBN 0-8053-5340-2.

[11] Davis, A. M. (1993). Software Requirements: Objects, Functions, and States, Prentice Hall.

[12] Boehm, B. W. (1981). Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall.

[13] Nakajo, T., Kume, H. (1991). A Case History Analysis of Software Error Cause-Effect Relationships. Transactions on Software Engineering, 17 (8) 830-838.

[14] Jacobson, I., Wei Ng, P., Spence, I. (2006). Enough process - let's do practices, *Journal of Object Technology*, 6 (6).