Project BTW: if you go, my advice to you

BTW: UFPE Summary Report For SCORE Competition

Clarissa Borba, João Henrique, Laís Xavier



1

Table of Contents

Introduction	3
Development process	3
Planning and management	
Requirements engineering	
Problem Statement	9
Software Requirements Specification	10
Prototyping	12
Mash-up engineering	13
Architecture	14
Detailed Design	16
Implementation	17
Lessons Learned	
References	19

Introduction

This is the summary report of the project BTW: if you go, my advice to you, for the SCORE Competition, to be held in ICSE 2009. This document summarizes the used methodology and the execution of the project, providing information for the SCORE committee evaluation. Part of this project was performed as a project for a software engineering course.

In the BTW project, we are asked to develop a route-planning system that allows community input. The proposal determines that, once defined a route, the user of the system can receive information about that route, posted by other user, to help in its planning. The advices might need to be filtered, to provide the user with relevant information about the place he pretends to visit. In summary, the system is an attempt to move beyond the official GIS information that might be provided by a government or private agency, and allow the traveling public to provide advice.

With this project we intended to put in practice novel techniques that are being proposed in academic literature, so that we could not only build the system but also learn in the process, besides this being a great opportunity to put in practice some of these techniques.

Development process

When reading the project description, we realized that it was well suited for an agentoriented approach, due to its characteristics of advice suggestion and information matching. There are a plenty of papers about recommender agents, like summarized in [13]. Using agent-oriented development was also a good learning opportunity and a possibility to get insights for further research on this kind of development.

There exist many agent-oriented development methodologies [8]. GAIA [5] and Tropos [4] are two popular approaches. Both methodologies could be used for the development of this project, but we choose to use Tropos mainly because of five factors: it is requirements-engineering oriented; it has a plenty of papers explaining and extending it (see [12] for some of them); its seminal paper is the fourth most cited paper, according to Science Direct listing in October 2008; the research group we take part was directly involved in the creation of Tropos; there is a well defined process to use the Tropos methodology. From these factors, the most important is the requirements-engineering orientation, since the Score committee made it clear that requirements is a main concern in this project, in their e-mail messages.

Since its creation, Tropos has evolved in two main branches: Tropos from Canada and Tropos from Trento. One of the members of our research group recently defined a development process that try to unify these branches, entitled U-Tropos [6]. This is the process that we used in our development.

Tropos has 5 disciplines, as follows:

- *Early Requirements* concerned with the understanding of a problem by studying an organizational setting.
- *Late Requirements* where the system-to-be is described within its operational environment, along with relevant functions and qualities.
- Architectural Project where the system's global architecture is defined in terms of subsystems, interconnected through data, control and other dependencies.
- *Detailed Project* where behavior of each architectural component is defined in further detail.
- *Implementation* were the detailed design specification must be followed step by step in order to implement the application and produce an executable release.

Management and Verification disciplines were added in U-Tropos to complement the development process. These 7 disciplines are linked together as showed in Figure 1. In the first phase, "Conception", is made the first definition of the system to be developed, where the project scope is defined. In the "Design" phase, the system architecture and the agents design are defined. In the third phase the system is coded, generating a running version of the system. In the "Deployment" phase, the system is made available to its final users. This includes testing, configuration and training.



Development Life Cycle

Figure 1 - Adapted by U-Tropos overview

Planning and management

Like any other process, it is necessary adaptations during U-Tropos instantiation, which in this case consists of defining a subset of activities to be performed in each iteration. The activities that we selected are listed in Table 1, Table 2, Table 3, Table 4 and Table 5. Each table presents an iteration adapted from U-Tropos. The Activities are

labeled with: ER - Early requirement; LR - Late Requirement; AD - Architectural Design; DD - Detailed Design; Dev – Development; VV - Verification and Validation and PM - Project Management. At the end of each iteration, the produced artifacts are validated by real stakeholders. Validating our results at each iteration, we aim to produce a system that is valuable to real users.

Table 1 - Iteration 1 - U-Tropos Instantiation for the project BTW

Iteration 1 – System Definition
PM1 - Project Management
PM1.1 - Define Project Plan
PM1.2 - Define Schedule
PM1.3 - Define Risk Mitigation Plan
PM1.4 – Planning iteration
ER1 - Requirement Elicitation
ER1.1 - Identify Stakeholders
ER1.2 - Identify Relationships between stakeholders
ER1.3 - Identify Detailed Activities of Stakeholders
ER2 - Analysis and Specification of Early
Requirements

Table 2 - Iteration 2 - U-Tropos Instantiation for the project BTW

Iteration 2 - System Requirements Analysis

PM1 - Project Management

PM1.4 - Planning iteration

LR1 - Discover System Requirements

LR1.1 - Elicit Dependencies of actors and system

LR1.2 - Identify how to obtain actors intentions

LR2 - System Requirements Analysis LR2.1 - Analyze Dependencies of system LR2.2 - Means-ends Analysis LR2.3 - Contribution Analysis

Dev1 – Prototyping Dev1.1 - Specify GUI Prototype

VV1 - Validation of Artifacts VV1.1 - Validate Proposed Models

Table 3 - Iteration 3 - U-Tropos Instantiation for the project BTW

Iteration 3 - System Architectural Design

PM1 - Project Management PM1.4 – Planning iteration

PA1 - Specify Organizational Model PA1.1 - Identify Roles

PA1.2 - Identify Dependencies Between Roles PA1.3 - Identify Organizational Norms

PA2 - Select Architectural Style

PA2.1 - Analyze Architectural Style

PA2.2 - Select an Architectural Style

PA₃ - Define an Attribution Model

PA3.1 - Group Roles in subgroups PA3.2 - Analyze Correlation between subgroups and architectural styles

PA4 - Configure Architecture PA4.1 - Mapping Subgroups to Components of Architectural Style

VV1 - Validation of Artifacts VV1.1 - Validate Proposed Models

Table 4 - Iteration 4 - U-Tropos Instantiation for the project BTW

Iteration 4 - Early Requirements Review

PM1 - Project Management

PM1.4 – Planning iteration

ER1 - Requirement Elicitation

ER1.1 - Identify Stakeholders

ER1.2 - Identify Relationships between stakeholders ER1.3 - Identify Detailed Activities of Stakeholders

ER2 - Analysis and Specification of Early
Requirements
ER2.1 - Analyze Actor dependencies
ER2.2 - Means-ends Analysis
ER2.3 - Contribution Analysis
VV1 - Validation of Artifacts
VV1.1 - Validate Proposed Models

LR1 - Discover System Requirements
LR1.1 - Elicit Dependencies of actors and system
LR2 - System Requirements Analysis

LR2.1 - Analyze Dependencies of system
VV1 - Validation of Artifacts
VV11 - Validate Proposed Models

Table 5 - Iteration 5 - U-Tropos Instantiation for the project BTW

Iteration 5 - Late Requirements Review

PM1 - Project Management

PM1.4 - Planning iteration

LR1 - Discover System Requirements

LR1.1 - Elicit Dependencies of actors and system

LR1.2 - Identify how to obtain actors intentions

LR2 - System Requirements Analysis

LR2.1 - Analyze Dependencies of system

LR2.2 - Means-ends Analysis

Table 6 - Iteration 6 - U-Tropos Instantiation for the project BTW

Iteration 6 - Architectural Design Review

PM1 - Project Management PM1.4 - Planning iteration

PA1 - Specify Organizational Model

- PA1.1 Identify Roles
- PA1.2 Identify Dependencies Between Roles
- PA1.3 Identify Organizational Norms

PA2 - Select Architectural Style

PA2.1 - Analyze Architectural Style PA2.2 - Select an Architectural Style

PA₃ - Define an Attribution Model PA3.1 - Group Roles in subgroups

LR2.3 - Contribution Analysis

Dev1 - Prototyping

Dev1.1 - Specify GUI Prototype

VV1 - Validation of Artifacts

VV1.1 - Validate Proposed Models

PA3.2 - Analyze Correlation between subgroups and architectural styles

PA₄ - Configure Architecture PA4.1 - Mapping Subgroups to Components of Architectural Style

VV1 - Validation of Artifacts VV1.1 - Validate Proposed Models

Table 7 - Iteration 7 - U-Tropos Instantiation for the project BTW

Iteration 7 - Detailed Design

PM1 - Project Management PM1.4 - Planning iteration

DD1 - Architectural Design Refining

DD1.1 - Mapping i* to UML

DD1.2 - Model Architecture

DD2 - Social Patterns Analysis DD2.1 - Analyze Actor Capabilities DD2.2 - Select Social Patterns for the system DD2.3 - Apply Social Patterns to Agents

VV1 - Validation of Artifacts VV1.1 - Validate Proposed Models

Table 8 - Iteration 8 - U-Tropos Instantiation for the project BTW

Iteration 8 - Development of First Prototype

PM1 - Project Management

PM1.4 - Planning iteration

Dev2 - Coding

Dev2.1 - Set up development environment

Dev2.2 - Implement Agents Dev2.3 - Generate release

VV1 - Validation of Artifacts

VV1.1 - Validate Proposed Models

VV1.2 - Validate Proposed GUI

VV1.3 - Validate Prototype Acceptance

VV2 - Unit Tests
VV2.1 - Specify Unit Test Suite
VV2.2 - Apply Unit Tests
VV2.3 - Fix Bugs in Units
VV3 - Integration Tests
VV3.1 - Specify Integration Tests
VV3.2 - Apply Integration Tests
VV3.3 - Fix Bugs in Components
VV4 – Acceptance Tests
VV4.1 - Specify Acceptance Tests
VV4.2 - Apply Acceptance Tests
VV _{4.3} - Fix Issues in the System

We planned iterations 4, 5 and 6 for review and refinement of the work done in the earlier iterations, so that we could work upon the professor of the software engineering course suggestions. In this way, the activities described in these iterations are not performed from scratch, but instead are just an improvement on the activities performed earlier.

The Table 9 shows the general schedule of the project. At the beginning of each iteration it was defined a detailed schedule for that iteration.

Iteration	Starting date	Ending date
1 - System Definition	04/09/2008	30/09/2008
2 - System Requirements Analysis	31/09/2008	20/10/2008
3 - System Architectural Design	21/10/2008	14/11/2008
Deadline: artifacts for the so	ftware engineeri	ng course
4 - Early Requirements Review	15/11/2008	29/11/2008
5 – Late Requirements Review	30/11/2008	14/12/2008
6 - Architectural Design Review	15/12/2008	28/12/2008
7 - Detailed Design	29/12/2008	14/01/2009
Deadline: summary report for SC	ORE competition	n (15/01/2009)
8 - Development of the First Prototype	16/01/2009	25/02/2009

Table 9 - Schedule of BTW Project

Our project plan document is based on the IEEE Standard for Software Project Management Plans, but once we are a small team we preferred to omit some of its sections. We particularly elaborated the risks management section, since we knew from the very beginning that our project was a risky one.

We had weekly tracking meetings, where we monitored the project progress and refined the planning for the following week. During these meetings, we checked if a new risk has raised, and if we needed to perform any action planned for preventing risks. Often, in these meetings, we realized that one of the activities expected to be performed were not performed, but these were always low priority ones and did not impact the overall planning.

Based on the high amount of effort we needed to carry out the project, we believe that if we had a team of 4 or 5 students we would have made the project with considerably less sacrifice.

Requirements engineering

In this section we briefly explain the work we made regarding requirements engineering, and then present the resultant models of the problem statement (early requirements) and of the software requirements specification (late requirements).

Requirements elicitation - We used literature analysis, interviews and competitor analysis techniques. In literature analysis, we read information available on Internet about traveling in general, as well as travelling for these specific groups: physical impaired travelers, sensorial impaired travelers and cognitive impaired travelers. For the last group, the main reference was the TREK ACTs Wheels [7].

The interviews were semi-structured narrative-episodic interviews, with the interviewees telling how they planned their trips and describing the trips themselves. We first made these interviews with three Brazilians, and then made it with some slight variations with 5 foreigners, by e-mail. We believe that this contact with real stakeholders was a key factor on our effort to build a system really good for the users, and not for ourselves.

The competitor analysis was realized analyzing features from 16 software products that share some features with BTW, both desktop and web-based. After the third iteration, we also used rapid prototyping for requirements elicitation.

Requirements modeling and documentation – Tropos requires goal models, which are usually modeled using i*. To generate the i* models, we used the Process Reengineering i* Methodology (PRiM) [3]. In this methodology, we first build detailed interaction scripts, which are a basis for creating the i* models, using the heuristics provided by the methodology. These scripts are generated both in early and in late requirements. The tool used for modeling was OME3[15], a stable, functional and free tool with support to i* modeling. The requirements document, which contains the models, is based on the IEEE-830 standard.

Requirements analysis – For requirements analysis, we used the Scenario-Based method proposed by Alistair Sutcliffe [14]. This method comprises the following steps: Goal analysis, Inbound events analysis, Characterize system output, Output requirements analysis, Social impact analysis and Stakeholder analysis. Besides, the derivation of requirement models into architectural models was also a valuable source of insights for requirements analysis.

Requirements validation – The requirements were validated by the same people who we interviewed. After the third iteration, we based our requirements validation on prototypes.

Requirements management – All members of the team were allowed to modify and update the artifacts, but one person was responsible for managing the requirements artifacts. We maintained traceability links between the late requirements model and the architecture model.

Problem Statement

This section present the problem statement defined in the requirement phase. In U-Tropos, the problem statement is made in the Early Requirement discipline. Here we used an organizational analysis to identify who are the important people and organizations that are related with trip planning. Based on documentation and interviews we identified actors that are interested in a trip planning. They could be persons such as a traveler, or organizations such as a transportation companies. We used i* models to represent these persons and organizations and their relationships. To obtain the i* models the interviews and documents analyzed were transcribed to Detailed Interaction Scenarios (DIS), which contains a fine-grain refinement of activities.



Figure 2 – SD Model of Early Requirements

The Figure 2 shows a Strategic Dependency (SD) model that represents the actors related with trip planning and their social context. The main actor of this model is the Traveler, that will plan and realize a trip. The Traveler is not capable of planning a trip without interacting with other actors. This interaction is modeled in terms of dependencies between the Traveler and other actors. The activities that the Traveler does, while planning and executing a trip, requires information and services provided by several different actors. The Traveler may interact with the following actors:

- Community Support A Traveler needs to interact with the community to obtain information about places or conditions which will help in planning process.
- Transportation If some part of a trip requires the motion from one place to another and cannot be done by foot, the Traveler may consider taking different transportation vehicles, like subway or bus.
- Internet Nowadays maps services and information about places are available in internet and the Traveler may use this information while planning a trip.

• Hosting Place - If a trip will requires a break, the Traveler can select a hosting place and schedule the stops in advance.

In addition, we identified that a Traveler can be of different types. A Traveler can be a Person with Cognitive Challenges (PwCC) that need more interaction and take care with details in the planning activity. Other type of Traveler is the Hitchhiker that plan and execute a travel without considering use of a private vehicle. The Cyclist is a Traveler that requires a specific transportation and routes that have support for bicycles. In short, we identified that the Traveler needs to interact with many actors to plan a trip. A traveler seeks to reduce the number of his dependencies and increase the reliability of his information. Thus, the BTW system can help in this point using advices as a way to provide information through collective knowledge.

The Strategic Rationale (SR) model details how each actor supports the goals, softgoals or tasks which depend on him. Each goal is decomposed with a means-end link, the tasks are decomposed with decomposition links and the softgoals are decomposed with contribution links.

Software Requirements Specification

The Figure 3 is a Strategic Dependency model of the organization, i.e., the actors that are within the project scope and their relationships, now including the system. In this model the system is represented by the BTW actor, which has impact over other agents. The included system impacts the organization by redirecting some dependencies from other actors to the BTW actor. E.g., formerly when a participant of this context wanted to publish or read information about a trip he needed to use several internet services for each kind of information. With the inclusion of system (the BTW actor), these services, which were represented in the model by Goals, could be grouped in a unique application. The BTW system is now responsible for satisfying these needs through a recommendation system.

The Strategic Rationale (SR) model represents the rationale that motivate the dependencies or that satisfy the dependencies. Thus, the SR model explains why a given dependency is necessary and how the actors pretend to work in order to satisfy that dependency.

The Figure 4 represents the SR model of the BTW system. This model shows how the BTW actor satisfies the need of the other actors. As the main objective of the system is to provide advices in a trip, the high level goal is *Trip advices be provided*. This goal is accomplished by the task *Provide Advising Service*, which is decomposed in sub-tasks that represent the main activities related with the recommendation mechanism. These sub-tasks are *Add Advice* and *Show Advices*. A task can also present sub-goals that need to be achieved, like in the *task Provide Advising Service*, where its sub-goal *Advice be Updated* can be achieved in tree ways. These sub-tasks are directly related with the dependencies of other actors. Moreover, all those sub-tasks are restricted by a constraint, which is the softgoal *Security*.



Figure 3 - SD Model of Late Requirements



Figure 4 - SR Model of BTW System

The goal *User Access be Controlled* is related with the handling of user information. This control is required to maintain information about preferences of user and increase the security of their information.

Other goal of the BTW System is *Map Be Handled*. This goal is related with the maps services that are required to locate the advices and show them in a map. The task *Provide Maps services* involves integration and adaptation of other internet services.

In order to simplify the representation in this model, some actors were grouped in two types of actor: those who give advices (*Advice Giver*), and those who receive advices (*Advice Receiver*).

Prototyping

We have decided to use the prototyping technique for two reasons. The first one is because it is an efficient technique to gather requirements from stakeholders. The second one is because it fulfills U-Tropos deficiency on proper dealing of usability, which is believed to be a critical factor for web-sites success.

We built throwaway paper prototypes of some parts of the system, and tested them with small subsets of users, focusing on qualitative feedback. The tests were performed focusing on 2 main user tasks of the system, which briefly are: Get Advice and Provide Advice. The prototypes and the tests were produced based in [1].

The paper prototypes are low fidelity prototypes. This kind of prototype enables the quick evolution of the system requirements and user interface. Each prototype was tested by 3 potential users. Each test was started with the test facilitator reading general instructions for the test user, and then reading the script of the task that the user needed to perform. Next, the user interacted with the prototype with a pen, in replacement of a computer mouse. Finally, he was asked to externalize his thoughts on the test. With the test finished, the test facilitator wrote down his observations about the test and the user comments. These annotations are used to provide insights for a new version of the prototypes.

The table 6 shows the user script of each prototype that was built. The user script contains the task that the test users had to perform during the usability tests. From one version of the prototype to another, there were slight modifications made in order to explore some particularities of the prototype, but each task always kept its focus, respectively in Get Advice and in Provide Advice.

Task	Version	User script
1	1	You are going to travel to Ceará. You must read information about
		the route from UFCE to Jacaré Praia Hotel.
1	2	You are going to attend to an event on Boa Viagem Praia Hotel.
		Discover accessibility information about the streets that you will pass by
		when going from UFPE to Boa Viagem Praia Hotel.

Table 10 - User scripts for usability tests

2	1	Imagine that you passed by Avenida Caxangá, in Recife/PE, and
		realized that it is a very dangerous way for bicycles. Insert that information
		about Avenida Caxangá in BTW. You are already logged in the system.
2	2	Imagine that you passed by Avenida Caxangá, in Recife/PE, and realized that
		it is a very dangerous way for bicycles. Insert that information about Avenida
		Caxangá in BTW.

One of the most important benefits of these tests with prototypes was realizing that sometimes the user may want to see information not about a route, but about a specific place. E.g., the user knows that to go from UFPE to Boa Viagem Praia Hotel he will need to pass by Domingos Ferreira Street, then he may just want to type "Domingos Ferreira", rather than typing its origin and destination.

The following picture shows the evolution from the first prototype version for task 1 to the second version, as an example. From one version to another, we changed some visual elements that were leading to confusion, and reorganized the display of information.



Figure 5 – Evolution of the task 1 prototype, from version 1 to 2

Mash-up engineering

We believe that mash-up engineering share some concepts with Commercial off-the-shelf (COTS) integration. So we used a COTS approach [18] to select which component we would use to provide geographical data and maps.

We compared 4 products that provide these data, regarding selection criteria and their requirements. The selection criteria were: Map providing, Route providing, Geographical reach, Availability, Documentation and Cost. As a result, we selected to use the Google Maps API.

We haven't found any paper, technical report or book that explains how to deal with non-agent components, like a third-party API, in Tropos. We dealt with this using an extra architecture diagram, as explained in the Architecture section, but we believe that further studies could be made to represent these components in the Tropos diagrams, and take them into account in Tropos steps and heuristics. I.e., these components could lead the selection of alternatives in the Detailed Design phase. So, addressing the question made in the BTW project description, "Does building a mashup in this way differ from the concepts taught in software engineering classes today?", our answer would be yes.

Architecture

In Tropos, the architecture is mainly a description of the agents that will compose the system and how they are interrelated. Since these agents will have to deal with other components which are not agents, we felt the need for defining an broader view architecture, like the one in [17], which is referenced in the BTW project description. The result is depicted in Figure 6. The User, either using a desktop or mobile interface, will interact with the system using Web pages. The Web pages will display information from and provide information to the Advices and users database, and the Recommendation agency. The Recommendation agency uses data from Advices and users database, as well as geographic data and maps from the Google Maps. The Recommendation agency is the part of the system that contains the agents, which architecture will be detailed in the remainder of this report.

The architecture in BTW system was built using the proposed SIRA framework [9], as defined in [6]. In the SIRA framework, the architecture is drawn from the functional and non-functional system requirements. The functional requirements (goals and tasks) identified in the late requirements discipline are used to identify an Organizational model and the non-functional requirements (softgoals) are used to identify an Architectural Style [11]. The techniques of Social network analysis [10] is applied and an Assignment model is set for the components mapping between these two models. When the Assignment model is run the architectural configuration is generated. The i* framework is used to model the system components and their interactions.



Figure 6 - General architecture of the BTW system

The organizational model is defined from the goals and tasks identified in the diagram of strategic rationale from Late requirements discipline. By this analysis five different roles were identified: *Advice Manager, Profile manager, Advice Publisher, Advice Recommender* and *Maps Manager*.

The architectural style is selected comparing the organizational styles defined in Tropos with the agents quality attributes. From the Late requirements model, we defined for desired qualities for the system agents: Security - Protocols and strategies for verifying authenticity for data sources captured by individual agents are an important concern; Adaptability - Agents may be required to adapt to modifications in their environment; Cooperativity - They must be able to coordinate with other entities to achieve a common purpose; Availability - Components that offer services to other agents must implicitly or explicitly guard against the interruption of offered services. The architectural styles that are better suited for these attributes are Pyramid and Joint-Venture. Since the Pyramid style uses a rigid hierarchical structure, we preferred to use the Joint-Venture style. Figure 7 shows the Organizational Model of the BTW System and the selected Architectural Style (Joint Venture).



Figure 7 - BTW Organizational Model and Architectural Style

The selected architectural style and organizational model are social networks that represent how the components (or roles) are related. Once the organizational model and architectural style were identified, we conducted a mapping among the components through the Social Networks Analysis, which measures the strength of relations between members in the group. It analyzes the centrality and structural similarity between these components. Upon completion of this analysis, an Assignment model is created and it determines the system Architectural configuration. The analysis resulted in the following similarities: the role Advice Publisher has similarity with the Joint management component; the role Information Collector has similarity to the role Secondary Partner component; the other roles from the organizational model are similar to the Principal partner, resulting in the architectural configuration showed in Figure 8.



Figure 8 – Architectural configuration

Detailed Design

The detailed design in TROPOS methodology involves the specification of each sub-system described in architectural design as models, with sufficient details to allow the system implementation. As defined in the process adopted in this project [6], we used the models proposed in [16].

This phase starts with the allocation of goals from the Late requirements Strategic Rationale model between the actors identified in the architecture, generating the Architecture Component-Goals model. Then we transform that diagram into a UML class diagram, using specific stereotypes as presented in Figure 10. Other five models, which are variants from this one, are built: Communication, Environmental, Intentional, and Rational models. Each model specifies a complementary part of system and will be used to guide the implementation phase. The communication model is an UML sequence diagram, and all the others are class diagrams.

With the models produced in detailed design we have a description of the agents, their goals, the environment where they will run, their communication protocols, and the plans and action that will be executed to achieve their goals.



Figure 10- Detailed Architectural model

Implementation

The implementation is based on web technology, including maps services and agent-oriented tools. We are using agent as the main abstraction in modeling and designing of the BTW project, then all non-agent part of the system, like third-party API and database servers, requires an agent which communicate with it. The advice giving involves a multi-criteria filtering of information to choose the most relevant advices. To solve this problem we are planning to use algorithms that handle this kind of problem, such as profile matching algorithms, correlation algorithms and forgetting functions [13].

The interactive nature of our project makes us consider the Web 2.0 solutions to develop the GUI and related interaction mechanisms. After a technical viability study, we choose some technologies that are already familiar to our team and that are free to use. These technologies are:

- JavaScript client script language that is necessary to access the maps API and user interaction;
- Java Server Page (JSP) script language to build GUI communication with server;

- Java and java related technologies for background processing and to access database servers and web services;
- PostgreSQL and Tomcat database server and web server;
- Java Agent Development Framework (JADE), a framework to develop multi-agents systems.

In our project, we assume that the agents are responsible by the server-side information processing. The integration between the agents and the web technologies is made in the application that runs in the web server. The agents developed with JADE runs in a container that communicates with the web server. The requests send to the web server are forwarded to the agents container, which responses with the processed data.

Lessons Learned

As we are part of a requirement engineering group we were very interested in interacting with real stakeholders to apply our skills in practice. However, we lost a precious time to identify and contact the stakeholders, what slowed down our project pace. Moreover, interact with real stakeholders is hard when they are not interested or they have little time to dedicate to the project.

It was interesting that each member of team made interviews with a different stakeholder and documented the interviews with DIS scenarios, because more information could be collected with different view points, which enriched the understanding of the problem. Although, we have to take into account that this choice require more time than if the same person makes all the interviews.

The BTW project could be developed with agile processes such as XP or SCRUM. However, we preferred to use U-Tropos to study pros and cons of this process in development of a real study case. U-Tropos is not agile and this became evident during this project. Using this unified process we spent more time preparing the development than we would be if we used agile methodologies. On the other hand, we produced artifacts that will be used to some researches and which we expect to help us improving the Tropos methodology in many ways. E. g., we can study decrease the need for artifacts in Tropos.

It was important for us to realize that not always what the user say that would be good, would really be good for him and, even more important, for other users. If we had accepted all suggestions that were raised during the tests with prototypes, our BTW system would become some kind of freaky system.

At this moment, the Tropos methodology has poor tool support. The tools just cover some parts of the methodology. Most models are refinements over the previous ones, and there is no tool that performs automatic transformations on them, or even tools that provide traceability links between the models elements. Thus, we found difficult to maintain traceability without tool support.

References

- 1. US. DEPARTMENT OF HEALTH AND HUMAN SERVICES. "Research-Based Web Design and Usability Guidelines". United States of America, 2006.
- 2. ALVES, C.; CASTRO, J. "CRE: A Systematic Method for COTS Components Selection". In proceedings of XV Simpósio Brasileiro de Engenharia de Software, 2001.
- 3. GRAU, G.; FRANCH, X.; MAIDEN, N. "A Goal-Based Round-Trip Method for System Development". In Proceedings of the 11th International Conference on Requirements Engineering: Foundations for Software Quality (REFSQ'05), 2005.
- 4. CASTRO, J.; KOLP, M.; MYLOPOULOS, J. "Towards requirements-driven information systems engineering: the Tropos project". In Information Systems Journal, 27, 6. 2002.
- 5. WOOLDRIDGE, M.; JENNINGS, N.; KINNY, D. "The Gaia Methodology for Agent-Oriented Analysis and Design". In Journal of Autonomous Agents and Multi-Agent Systems, 3. 2000.
- 6. SILVA, M. "U-TROPOS: uma proposta de processo unificado para apoiar o desenvolvimento de software orientado a agentes". M.Sc. dissertation, Federal University of Pernambuco, 2008.
- 7. LEMONCELLO, R., SOHLBERG, M. M., & FICKAS, S. (2007, November). Activities of Community Transportation (ACTs): A Model of Community Navigation. Poster presented at the annual convention of the American Speech-Language-Hearing Association: Boston, MA.
- 8. HENDERSON-SELLERS, B.; GIORGINI, P. "Agent-Oriented Methodologies". Published by: Idea Group, Inc. 2005.
- 9. BASTOS, L. R. D.: Integration of System Requirements and Multi-Agent Software Architecture. Tese de doutorado, Universidade Federal de Pernambuco, Centro de Informática (2005).
- HANNEMAN, R.A., RIDDLE, M.: Introduction to social network methods. Riverside, CA: University of California, Riverside (published in digital form at http://faculty.ucr.edu/~hanneman/) (2005)
- 11. KOLP, M.; GIORGINI, P.; MYLOPOULOS, J. "Multi-Agents Architectures as Organizational Structures". In Journal of Autonomous Agents and Multi-Agent (JAAMAS), 13(1):3-25, Springer. 2006.
- 12. GIORGINE, P., MYLOPOULOS, J., PENSERINI, L., PERINI, A., SUSI, A. Tropos at the Age of Eight: On-going Research at FBK, UniTN and UT. iStar 2008: 83-89
- 13. MONTANER, M., LÓPEZ, B., DE LA ROSA, J. A Taxonomy of Recommender Agents on the Internet. In Artificial Intelligence Review, June 2003.
- 14. SUTCLIFFE, A. Scenario-Based Requirements Analysis. Requirements Engineering, 3-1, 1998.
- 15. YU, E, YU, Y. Organization Modelling Environment tool. Available for download in http://www.cs.toronto.edu/km/ome/
- 16. SILVA, C. T. L. L. Separating Crosscutting Concerns in Agent Oriented Detailed Design: The Social Patterns Case, PhD Thesis, 2007.
- 17. THANG, M. D., DIMITROVA, V., DJEMAME, K. Personalised Mashups: Opportunities and Challenges for User Modelling. In Proceedings of the 11th international conference on User Modeling, Springer-Verlag, 2007.
- 18. CARVALLO, J. P., FRANCH, X., QUER, C. Determining Criteria for Selecting Software Components: Lessons Learned. In IEEE Software. IEEE Computer Society, 2007.