# PASSI Methodology in the Design of Software Framework: A Study Case of the Passenger Transportation Enterprise

Daniel Cabrera-Paniagua and Claudio Cubillos

Pontificia Universidad Católica de Valparaíso, Escuela de Ingeniería Informática, Av. Brasil 2241, Valparaíso, Chile
daniel.cabrerap@gmail.com, claudio.cubillos@ucv.cl

**Abstract.** This paper presents a practical experience on the use of the PASSI methodology in conjunction with a general framework development process, in obtaining a software framework for a virtual enterprise for passenger transportation. In addition to background information on each of the topics discussed, diverse PASSI artifacts complemented with notational elements drawn from UML-F are shown. In addition, the experience on the use of PASSI for framework development is provided.

**Keywords:** PASSI, Framework Development Process, Agents, Passenger Transportation.

## 1 Introduction

The agent paradigm constitutes a significant forward step in the future of systems development, similar to a revolution in the software area [5]. Until a few years ago, the development of multiagent systems considered very few the use of software engineering techniques [9]. The development of these systems was based on ad hoc procedures, which allowed a high degree of flexibility to the characteristics of the project tackled. However, there were innumerable problems, which mainly are summarized in deficiencies of utilization of available resources, and precarious levels of quality, since the formal processes of testing and quality assurance did not exist. Therein lays the importance of using software engineering techniques in the systems development, and in this particular case, its application to the development of multiagent systems. While this offers various direct benefits, it should be noted that the level of the final obtained result depends, among others, on the quality of the software development process used.

The present work describes the application experience with PASSI methodology [1] in the design of a software framework for the domain of passenger transportation. Because the project involved a software framework, it has been considered the use of a general process for framework development along with PASSI. Additionally, we have incorporated some elements offered by a non-standard UML profile called UML-F [7], devoted to the development of object-oriented software frameworks.

M. Luck and J.J. Gomez-Sanz (Eds.): AOSE 2008, LNCS 5386, pp. 213–227, 2009.

The novelty of our work relies on: 1) show the practical experience of using PASSI in obtaining a software framework and 2) An analysis of the evidence obtained through the study case.

## 2 Related Work

In some European countries local authorities have introduced flexible public transport and demand responsive services that have proved to be most popular among users, and thus support the retention of citizens in their every day environment. Pilot experiences for DRT systems were developed during projects such as SAMPO [10], SIPTS [11], and FAMS [13]. Regarding examples of MAS applied to ITS, just to mention some initiatives, Ferreira et al. [14] presented a multi-agent decentralized strategy where each agent was in charge of managing the signals of an intersection and optimized an index based on its local state and "opinions" coming from adjacent agents. In 2002, Cai and Song [12] introduced a traffic control model with MAS, in which a more flexible agent self-control framework was described and a multi-agent negotiating strategy was conceived.

This work represents the continuity of a past research in this transportation domain [6] [17], concerning the development of an agent system for passenger transportation for a single operator under a demand-responsive scenario.

## 3 The Framework Development Process

As mentioned above, the methodology to use for developing software is of vital importance. Here are reviewed those aspects relating to software development process adopted in the present work.

### 3.1 The PASSI Methodology

PASSI (Process for Agent Societies Specification and Implementation) is a step-by-step methodology for designing and developing multiagent systems. PASSI integrates design models and concepts from both OO software engineering and artificial intelligence approaches using the UML notation. Figure 1 shows the PASSI methodology, which is made up of five models plus twelve steps in the process of building a multi-agent system. For a more detailed description on the different steps please refer to [1]. According to the figure, and considering as an exception the phase of "Agent Implementation Model", highlights the prominent sequentiality of the methodology. This contrasts with the iterative nature of the development of software frameworks. That is why PASSI has been used within an overall process of software framework development.

### 3.2 Process of Developing Software Frameworks

Historically, one of the most important topics taken into account in the area of software development is its reuse. Software reuse allows reaching a faster software development while promising a higher quality level. In this sense, software

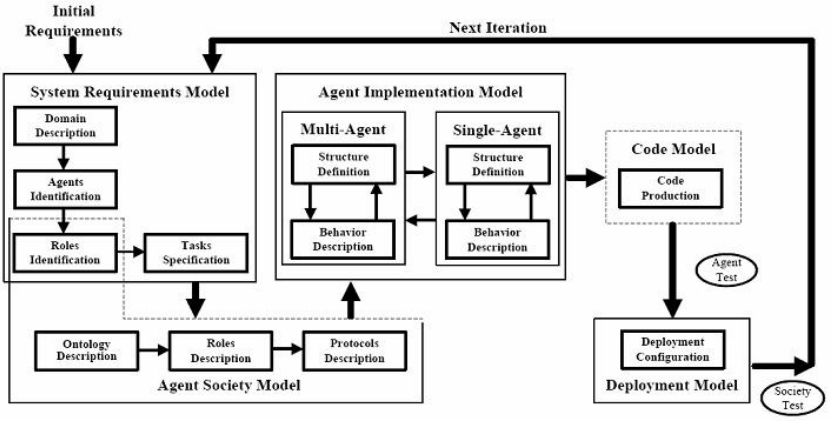


**Fig. 1.** The PASSI methodology

frameworks, one of the alternatives to carry out reuse, have gained considerable popularity in both the industry and academia. According to [8], *"a framework is a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuses at a larger granularity than classes"*.

Certainly, the software systems development is not an easy task, given mainly by the large number of variables and constraints involved. And in this respect, to develop reusable software system in time is much more difficult. A flexible software system must meet its requirements, while confining the solution for a wide range of future problems. That is why it is necessary to have the maximum possible aid elements. One is the process of developing frameworks, in a systematic way that will get a higher quality framework.

In the literature exists several proposals about development process for the obtaining of software frameworks [2] [19]. The considered process for framework development corresponds to the one presented in [3], where some stages are identified that should be part of an overall process of software framework development (see Figure 2). The steps involved in this approach are [3]:

Analysis of the problem domain. This is performed systematically or through development of one or a few applications in the domain and the key abstractions are identified. The first version of the framework is developed utilizing the key abstractions found.

One or possibly a few applications are developed based on the framework. This is the testing activity of the framework. Testing a framework to see if it is reusable is the same activity as developing an application based on the framework. Problems when using the framework in the development of the applications are captured and solved in the next version of the framework. After repeating this cycle a number of times the framework has reached an acceptable maturity level and can be released for multi-user reuse in the organization. Is important to say that the success of development process of a framework depends on the

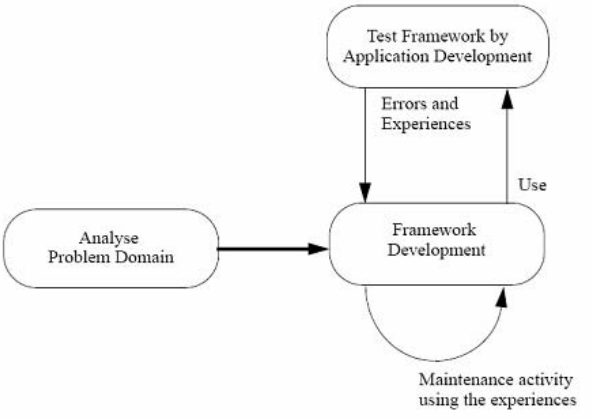


**Fig. 2.** The General Framework Development Process

experience of the organization in the problem domain the framework addresses. A more experienced organization can select a more advanced development process since they will have fewer problems with the problem domain.

The software frameworks raise that it is possible achieve reuse, through the development of software configurable architectures, based primarily on identifying points of fixed nature, and points of variability. Determine the points of the architecture that should be variable and the points that are not is the most difficult task in developing software frameworks. So, the general framework process development mentioned previously indicates, as a basis, an iterative development process, considering the experience and feedback to the fullest possible way. Hence it is mentioned that a software framework never ends developing it. Over time, sooner or later it is necessary to make modifications, to a greater or lesser degree, to initial architecture reached.

Particularly at present work, the main objective consists in development an agent architecture for a specific problem domain, the passenger transportation enterprise. But must not lose sight of that the architecture should offer a flexible capacity, in the sense of dispose obtaining several final systems from it. In other words, is necessary achieve a software framework for the specific problem domain addressed. For this reason, the framework design is supported by the use of two processes as a whole. On the one hand, there is a general framework development process, which roughly indicates the overall development line, namely: an initial analysis of the problem domain, actors and systems involved, and so on; development of the framework itself (obtaining the software artifacts) and validation, through an application built by extending the framework.

On the other hand, lays PASSI, a multiagent systems development methodology, aiding the development of complex, multi-party, distribute and heterogeneous software systems based in the agent concept as base modeling unit.

### 3.3 The UML-F Profile

The UML-F profile represents an important alternative in the development of software frameworks, because it formalizes aspects not covered by the UML standard. Some of the most important features of the UML-F profile are: to provide elements of notation to adequately document well-known design patterns; is built on the standard UML, that is, the extensions generated can be defined on the basis of extension mechanisms which in UML already exist; is, in itself, the medium that allows a direct way to document any pattern framework. The PASSI methodology was designed for the development of agents systems and not precisely for the developing of agent-oriented software frameworks. For the same reason, the UML-F profile was used in the development of some artifacts, with the aim to fill this gap. Please refer to [7] for more details on the UML-F notation.

## 4 The Study Case: Passenger Transportation Enterprise

The Intelligent Transport Systems (ITS) have been attracting interest of the transport professionals, the automotive industry and governments around the world. The ITS aim at the development of the road infrastructure (for example, ways) and to integrate them together with the persons and vehicles by means of advanced technologies of integration, from several research areas.

Regarding the public transport domain, in the last years the Demand Responsive Transport (DRT) services have risen in popularity. A DRT is understandable as a component of a long chain of inter modal service, delivering local and complementary mobility to other conventional transport means, such as fixed-line buses and trains. However, geographical coverage problems among transport operators services, difference in the volume and quality of handled information and, in general, a fragmentation in the transport service provision, gives origin to problems that range from wrong evaluations coming from state-regulatory entities, up to direct problems with the system final users, which definitively results in a poor quality of service. For these reasons in many cases the solution implies a better integration and coordination of the diverse parties, leveraging the concept of virtual enterprise.

### 4.1 Virtual Enterprise

A virtual enterprise is a cooperation network of legally independent companies, which are quickly united and mainly contributed their basic competences in sequence to exploit a specific business opportunity. In general terms, the life cycle of a virtual enterprise is marked by four phases, that go from the identification, evaluation and selection of business opportunities, to the selection of partners to conform the virtual enterprise; later, a phase of operation, in where the business opportunity is exploit; and a phase associated at the end of the virtual enterprise, with the corresponding separation of assets. In general, virtual enterprises are applicable to all those domains where it is possible to conceive a collaboration of

different companies or entities, taking as a benchmark to reach certain goals for themselves while using information technologies. In this sense, we may think in a virtual transportation enterprise, with an increased level of adaptability to the offer, considered the variability in the levels of existing demand. With this, and under new business opportunities, the virtual transportation enterprise adapts its structure to meet the existing demand.

### 4.2 Transport Requirements

In a complex system like the passengers transportation one, there are many users or actors who have a direct interest in the commercial, social and infrastructure impacts. The actors considered in a DRT service correspond to:

*User*: Represents the end user of the passengers transport system. The user has the faculty to make request of transport (with its respective conditions), as well as to indicate some problem that affects to him and that has incidence in the concretion of the trip requested.

*Transport Operator*: Represents a transport company within the system. A transport operator can correspond to an only person (even handling to she herself a vehicle, without delegating that responsibility in a conductor), or also correspond to a company composed by multiple vehicles (fleet of vehicles). The virtual transportation enterprise is conformed by manifold operators.

*VE Administrator*: The Virtual Enterprise Administrator represents the central administration of the virtual transportation enterprise. Its faculties are related to affiliation control of new transport operators into the virtual transportation enterprise and its permanence, and taking action when extern events to virtual transportation enterprise happen, without restricting the future allocation of other responsibilities.

*Driver*: This is a vehicle driver belonging to a transport operator. Receives the itinerary to follow, and can notify problems or indications to meet your itinerary (for example, report a delay in the time of encounter with a passenger).

*Government Entity*: Is a governmental organization with regulating or control faculties, which guard current legislation and that service contracts are fulfilled.

*Active Destination*: Represents a frequent destiny within total set of existing destinations. An active destination can make the virtual transportation enterprise see a necessity or a business opportunity available; as well as indicate problems associated to the same transport service, like a loss in the quality of service, or restrictions on the operation.

*Traffic Information System*: Represents an external information system that gives information on present traffic conditions, collisions, among others.

*VE - Customers Manager System*: Controls the profile of each user of the virtual transportation enterprise. The life of each user (related to the virtual transportation enterprise) is recorded in the first instance, with purely operational purposes, which does not prevent in the future can use this information for strategic purposes.

*VE - Transaction System*: This system controls all transportation requests completed, the requests are under way, and even those that have been canceled for various reasons, including all information existing in the request for transport, and the service characteristics offered by the virtual transportation enterprise.

*VE - Affiliates Enterprises Management System*: This system manages the life cycle of the virtual enterprise, from transport operators incorporation until the separation of transport operator from the virtual transportation enterprise.

*TO - Fleet Management System*: This system is responsible for managing the vehicles that comprise the transport operator fleet. This system depends directly on the transport operator.

*TO Solver*: This system has the task to optimize transportation operations (planning and scheduling) using a solver and/or heuristic software, on which are scheduled trips to perform for each vehicle.

## 5 Multiagent Framework Design

In this section, the agent framework artifacts are depicted following the PASSI steps. The first diagram presented shows a portion from the Agent Identification Diagram (see Figure 3), which is framed within the first stage of the PASSI methodology, corresponding to the System Requirements Model.

This diagram takes as starting point the description of UML use-cases, offering a general view of all the functionality provided by the system and in addition, it incorporates a grouping of use-cases for each agent identified within the system in order to visualize the responsibility level that each of the agents has regarding the system. The generation of diagrams is given on the basis of the use of a graphical tool available for PASSI, called PASSI Toolkit [4].

The UserAgent is who represents within the system the interests of the transport service user. It administers the transport preferences user, and manages his service requests. For this reason, it establishes communication with the

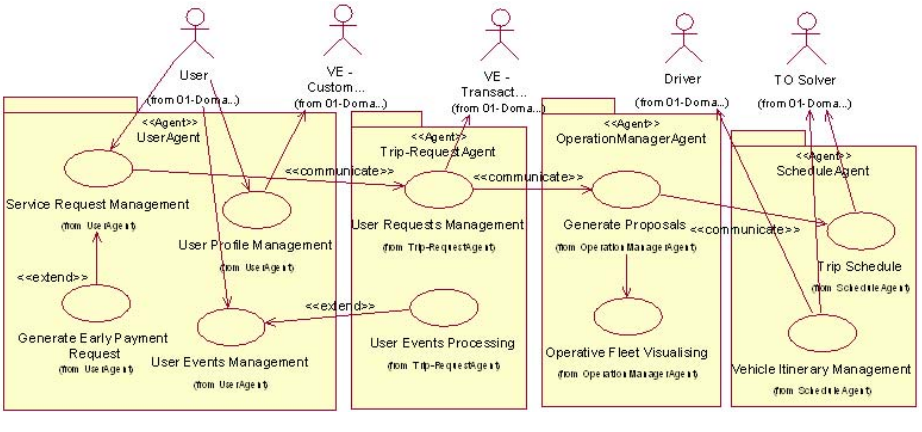


**Fig. 3.** Agent Identification Diagram

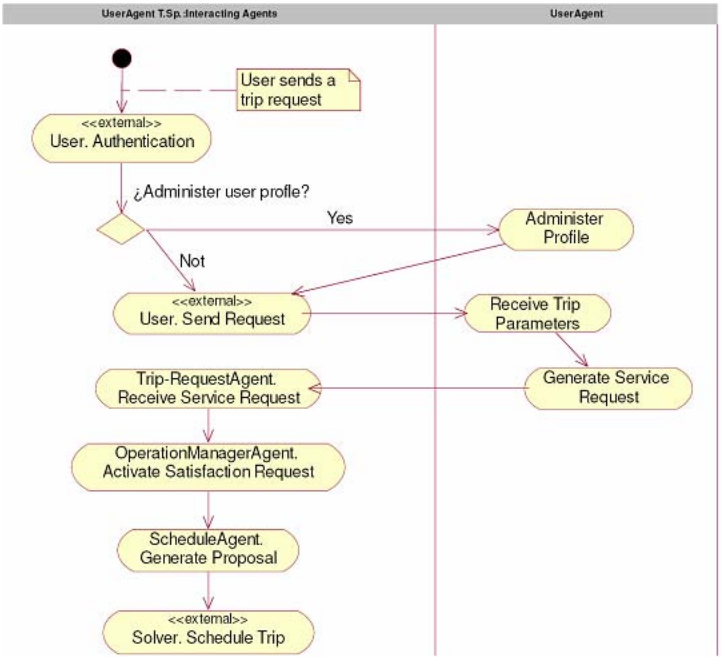


**Fig. 4.** Task Specification of the UserAgent

Trip-RequestAgent. Also, it allows the user the generation of an advanced-payment request, considering for this several means to carry it out. It can also communicate problems (called "events"), for example, report as late for meeting with the transport vehicle assigned for the trip. Each event is recorded in a single list for each user, and is sent to PlannerAgent for administration.

The Trip-RequestAgent manages the request of trips emitted by the user of the transport service, and maintains a registry of the pending requests. It sends the requests for its processing to the OperatorManagerAgent, receiving the proposals generated for each conducted request. Later, is recorded in the VE - Transactions System, the received request and the vehicle identifier.

The OperatorManagerAgent receives users trip requests, and active mechanisms for transport operators affiliated to the virtual enterprise attempt to generate an offer to the request. For this, knows the vehicles available at all times, and in operational service.

The ScheduleAgent verifies for a particular vehicle if it fulfills the conditions specified on a requested trip (user conditions, conditions of the virtual enterprise, or conditions caused by external events), checking its itinerary obtained from the information system of the transport operator. Considering the feasibility verification, a proposal or a declination takes place.

Figure 4 shows a portion from Task Specification Diagram for the agent User-Agent, where a user of the transportation system sends a service request.

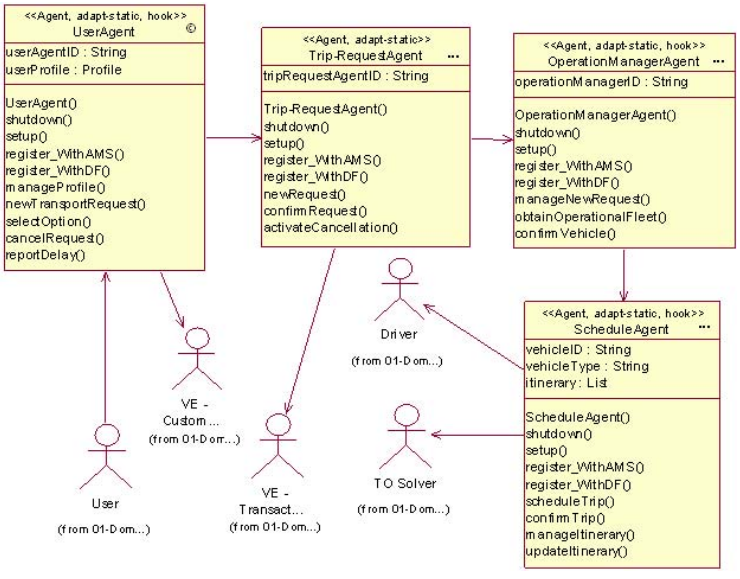


**Fig. 5.** Multiagent Structure Definition Diagram

The send of a transport request involves indicating a set of parameters on the trip, which are clustered: places (origin and destination), schedules (hour meeting at the place of origin, and arrival time at destination), and trip preferences. These preferences are managed through a user profile, which can vary over time. After having details of the travel request, it is received by the agent OperationManagerAgent, which establishes the operational fleet and makes a broad call for sending proposals to meet the request. Each vehicle (represented by the agent ScheduleAgent) tries to schedule the trip requested within their itinerary.

The Figure 5 shows a portion from diagram of the phase of Agent Implementation Model,which is the Multiagent Structure Definition. It is possible to view all actors within the defined architecture in development, and its relationship with the various agents, and the identifying the transactions related to each of them. The classes identified in the figure with the symbol "...", indicates that have not yet been established all internal elements of them (both attributes and methods). On the other hand, the classes with the symbol "©" are those that their methods and attributes shown are actually all the ones the class possesses. The stereotype <<agent>> indicates that the classes are agents, and the stereotype <<adap-static>> denotes those classes that may be subject to changes, but only changes at the design phase (at runtime is not possible to observe changes in its internal structure). The stereotype <<hook>> indicates that the class has at least one method of type "hot spot", that is, their characteristics depends on each particular implementation derived from the model defined.

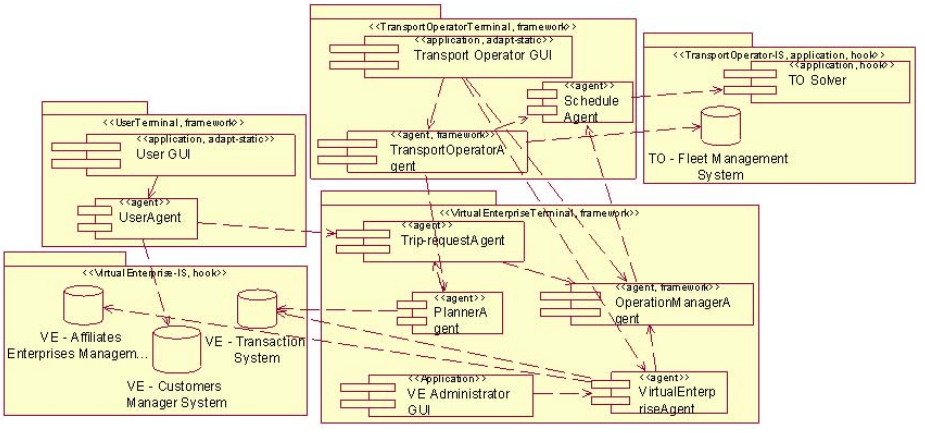


**Fig. 6.** Deployment/Component hybrid diagram of the framework

Next, a deployment/component hybrid diagram shows a portion of the general framework architecture (see Figure 6). Some packages have the UML-F stereotype <<framework>>. This means that these components are owned solely to the framework architecture. On the other hand, there are some packages that have the stereotype <<application>>. This stereotype indicates that these elements do not belong directly to the framework architecture, but are external components to framework, related somehow to it.

This architecture allows each transport operator affiliated to the virtual transportation enterprise to control at any moment the status of its operative fleets, as well as to administer all the information of their own information systems, in such a way that independence between the different transport operators stays in the operative scope. Each transport operator has his own mechanism for allocation and control of the itineraries for the different vehicles conforming its fleet, having the virtual transportation enterprise the only responsibility of receiving trip requests and the assignment of these requests to the vehicle that constituted the most attractive provider for the service user.

The Figure 7 shows a portion code of the UserAgent agent. This code is part of a functional prototype that is currently under development and is being tested with Solomons benchmark data sets for VRP and specially adapted for the passenger transportation problem.

The objective sought to develop a functional prototype is get some measure of how architecture raised behaves in a context closer to reality, so that in the future decide to develop applications from it, in a real context. The development of a functional prototype of large-scale, also demand efforts and resources to the same extent. That is why it is considered a limited scenario in the functional prototype.

The agents considered in the prototype for those who are directly involved in the receipt and administration of transport requests: UserAgent, ScheduleAgent,

```
public void setup(){

    //Generating the Controller Agent
    System.out.println("Yhe agent '"+getAID().getName()+"' is ready!");

    SequentialBehaviour sbObj = new SequentialBehaviour();

    //Adding subBehaviours
    sbObj.addSubBehaviour(new getUsersProfiles(this));
    sbObj.addSubBehaviour(new getRequests(this));

    sbObj.addSubBehaviour(new getScheduleAgent(this));
    sbObj.addSubBehaviour(new getOperationManagerAgent(this));
    sbObj.addSubBehaviour(new getTripRequestAgent(this));
    sbObj.addSubBehaviour(new getUserAgent(this));

    addBehaviour(sbObj);

}
```

**Fig. 7.** Code from the Controller Agent

```
public void action(){
    try{
        ACLMessage message = new ACLMessage(ACLMessage.INFORM);
        message.addReceiver(new AID(tripReqAgentParterID, AID.ISLOCALNAME));
        message.setContentObject(userRequest);
        message.setConversationId("satisfacer-peticion");
        message.setReplyWith("message"+System.currentTimeMillis());
        myAgent.send(message);

        MessageTemplate mt = MessageTemplate.and(MessageTemplate.
                             MatchConversationId("satisfacer-peticion"),
                             MessageTemplate.MatchInReplyTo
                             (message.getReplyWith()));
    }catch(Exception e){
        System.out.println("Problem in UserAgent: "+e);
    }
}
```

**Fig. 8.** Code from the UserAgent agent

Trip-RequestAgent, and OperationManagerAgent. In addition to the previously agents, an agent is incorporated during the prototype performance. This agent is called Controller (see Figure 8).

The Controller Agent performs certain tasks: Get the User Profile of each user's transportation system considered within the prototype; obtain transportation requests of each of the users of transport system (from an XML file); obtain the properties of each vehicles considered within the prototype (from an XML file), and generate the instances of agent ScheduleAgent with such information; generate the instances of agent UserAgent, including both the User Profile as the transportation request associated with each user; and finally, generate the instances of the remaining agents.

```
<ProfileTwo>
  <UserName>Aquiles Baeza</UserName>
  <ProfileID>002</ProfileID>
  <VehicleType>van</VehicleType>
  <SharingVehicle>true</SharingVehicle>
  <MaxTimeArrive>10</MaxTimeArrive>
  <AirConditioning>true</AirConditioning>
  <BicycleRack>false</BicycleRack>
  <WheelchairRack>true</WheelchairRack>
  <ReadingLight>false</ReadingLight>
  <WC>false</WC>
  <DVDPlayer>true</DVDPlayer>
</ProfileTwo>
```

**Fig. 9.** XML Code from an user profile

The prototype code being developed by Jade Platform [16]. The Jade platform offers some assurances of stability for applications developed with its technology, because it has the backing of major companies, and is implemented on Java technology, an industry technology consolidated around the world.

The information from each user profile, applications for transport, vehicles and properties, are obtained by the use of XML files (see Figure 9). The feed prototype based on XML files replaced the inclusion of real people and systems that deliver this information in a real context.

The prototype takes as its bases the work developed by Cubillos et al., which is based in a work developed in the year 1986 by Jaw et al. [18].

As mentioned, the scope of the current prototype is limited. Mainly, they are looking to get some kind of feedback on the overall architecture raised, as well as the entities included in it, and business processes defined. In this sense, has completed implementation of all actors involved in the scenario described, by subtracting complete the development of the component called TO Solver, associated with the node TransportOperator-IS. At present, prototype does not incorporate a mechanism for scheduling the requests of trip received to the various vehicles available within the virtual transportation enterprise, because the component TO Solver is currently in development.

## 6 Analyzing PASSI

Based on the experience with the PASSI methodology and from the present study case, it is possible to make some analysis on PASSI appropriateness. First, its use provides a traceability of requirements along its phases, through the obtaining of its various artifacts. This allows easily identify and isolate any problems that exist within the system models, and the same way, more accurately verify that the requirements tackled in the early stages of the methodology are adequately represented in the final solution.

On the other hand, the early identification of actors, agents and functionality associated with each agent is of vital importance, since it allows the work team to have an overview but comprehensive system development. Fundamentally,

this is reflected in having clarity about which agents will be integrated into the system, without needing to think about elements to be addressed in the future, as is its implementation technology. The tasks assignment to each agent complements the above mentioned, giving an overview of the features that are the responsibility of each agent.

The multiagent structure definition diagram describes agent classes involved in the system. These classes have a direct relationship with the agents identified in the first phase of PASSI. At this level, it is possible to observe attributes and behaviors in each agent, which gives a closer view to the final implementation. Automatic generation of this diagram through the PASSI Toolkit contributes significantly to the maintenance of consistency during systems development. In view of the above, the PASSI methodology constitutes an important guide in the development of systems based on agent technology. However, it also suffers from some elements that, if incorporated, would significantly improve the PASSI adoption in industry and academia. For example, it does not engage the user explicitly within the development process, for example, the inclusion of the final users from the initial steps, in order to guide the development. In the domain tackled in the present work, this has relevance, because the passenger transport domain considers direct participation of various human actors with the information system. Therefore, it becomes necessary to incorporate the user, for example, in some stage of usability tests on user interfaces.

Also, although PASSI incorporates the "testing idea" (both individual agent and overall level of the multiagent society), it does not explicit how carry them out. This lack of guidance in the testing reveals perhaps more important, as reflects the general lack of quality assurance processes. Anyway, it is necessary to mention that the Agile PASSI methodology [15] incorporates formally this step into the development process.

On the same line, a possible future work is to develop a proposal of process for the development of multiagent systems, but incorporating project management tasks, e.g. risk management, quality assurance, among others. With this, we can achieve a major improvement in agent software projects at industrial level, while complementing its existing level of use in the academic environment.

Finally, as regards the framework development and PASSI, it is possible to say that it is necessary to adapt PASSI at various points, in order to accommodate the iterative nature of the framework development process. Although it is iterative, the period of time required to complete a full PASSI iteration requires considerable effort.

Therefore, on some occasions it was necessary to return quickly to any past stage of PASSI, in order to correct and stabilize some requirements not entirely clear, breaking this sequentiality established by PASSI. Although this action have a cost (because stops the process, apply the necessary improvements, and finally, check the consistency between the different diagrams), it is better than waiting until the last steps of development process and start a new iteration just to begin to apply the corrections needed. For example on some occasions different problems were discovered in the step of Roles Identification and was

necessary to return directly to Agent Identification Diagram, breaking the natural sequentiality of PASSI.

## 7 Conclusions

A practical experience on the use of the PASSI methodology in conjunction with a general framework development process has been achieved. An analysis on the results obtained from the use of PASSI has been exposed. It is worth noting that PASSI is not a complete development methodology of software projects, it lacks certain stages and some generic software project artifacts. Future work is devoted principally to gather more background on the advantages and challenges offered by PASSI methodology to be used in obtaining agent-oriented software frameworks, in order to make any specific proposal to extent the obtained results.

## Acknowledgments

This work has been partially funded by the Pontifical Catholic University of Valparaíso (www.pucv.cl) through project No. 037.215/2008 "Collaborative Systems" Nucleus Project and by CONICYT through FONDECYT research grant No 11080284.

## References

1. Burrafato, P., Cossentino, M.: Designing a multiagent solution for a bookstore with the passi methodology. In: Fourth International Bi-Conference Workshop on AgentOriented Information Systems, AOIS 2002 (2002)
2. Johnson, R.: How to Design Frameworks. In: Tutorial Notes, 8th Conference on Object-Oriented Programming Systems, Languages and Applications, Washington, USA (1993)
3. Mattsson, M.: Object-Oriented Frameworks: A Survey of Methodological Issues. Technical Report 96-167, Dept. of Software Eng. and Computer Science, University of Karlskrona/Ronneby
4. PASSI Toolkit (PTK), `http://sourceforge.net/projects/ptk`
5. Jennings, N.: On agent-based software engineering. Artificial Intelligence 117, 277–296 (2000)
6. Cubillos, C., Guidi-Polanco, F.: An Agent Solution to Flexible Planning and Scheduling of Passenger Trips. IFIP AI 217, 355–364 (2006)
7. Fontoura, M., Pree, W., Rumpe, B.: The UML Profile Framework Architectures. Addison Wesley, Reading (2000)
8. Johnson, R., Foote, B.: Designing reusable classes. Journal of Object-Oriented Programming 1(2), 22–35 (1988)
9. Gomez, J.: Metodologías para el diseño de sistemas multiagente. Revista Iberoamericana de Inteligencia Artificial 18, 51–64 (2003)
10. SAMPO TR1046 - Systems for Advanced Management of Public Transport Operations, `http://www.cordis.lu/telematics/tap_transport/research/projects/sampo.html`

11. SIPTS - TEN45607 - Services for Intelligent Public Transport Systems, http://www.novacall.fi/sipts/e_default.htm
12. Cai, Z.H., Song, J.Y.: Model of Road Traffic Flow Control based on Multi-agent. Journal of Highway and Transportation Research and Development 19(2), 105–109 (2002)
13. FAMS - Flexible Agency for Collective Demand Responsive Services. IST-2001-34347, http://www.famsweb.com
14. Ferreira, E.D., Subrahmanian, E.: Intelligent Agens in Decentralized Traffic Control. In: IEEE Intelligent Transportation Systems Conference Proceedings, USA, August 2001, pp. 705–709 (2001)
15. Chella, A., Cossentino, M., Sabatucci, L., Seidita, V.: Agile PASSI: An agile process for designing agents. Journal of Computer Systems: Science and Engineering 21(2) (2006)
16. JADE: Java Agent Development Framework, http://jade.tilab.com
17. Cubillos, C., Guidi-Polanco, F., Demartini, C.: Towards a Virtual Enterprise for Passenger Transportation Using Agents. In: Sixth IFIP Working Conference on Virtual Enterprises, Valencia, Spain, vol. 186, pp. 569–576 (2005) ISBN 978-0-387-28259-6
18. Jaw, J., Odoni, A.R., Psaraftis, H.N., Wilson, N.H.M.: A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. Transportation Research B 20B, 243–257 (1986)
19. Wilson, D., Wilson, S.: Writing frameworks - capturing your expertise about a problem domain. In: Tutorial notes, The 8th Conference on Object-Oriented Programming Systems, Languages and Applications, Washington (1993)