University of Crete
Computer Science Department

Semantics-based User Tracking and Dynamic Re-Planning for a Contextual Guide

Emmanouil G. Nikoloudakis

Master's Thesis

Heraklion, June 2008

ABSTRACT

Research activity in the field of assisted human navigation during the past few years has been increasing drastically. There exist, and will appear even more, areas where electronically assisted navigation is becoming an important part of Ubiquitous System solution. Already, electronic navigation devices have strongly influenced the procedure of human wayfinding, especially with the aid of GPS technology for outdoor environments. However, examining the deployed indoor navigation systems, it is evident that opportunities for innovative research are available, particularly in the user interaction and tracking procedures of the navigation process. Taking advantage of these opportunities, the original motivation is to create a semantics-based Contextual Guide for indoor environments, designed for pedestrian navigation in accordance with the users' physical and mental profile. The Contextual Guide combines expertise from three different research areas. The first area deals with user profile management while the second is focused on spatial representation and path finding techniques. Finally, the third research area, which represents this thesis' subject, deals with user tracking during the navigation process. Specifically, the system is responsible for observing user's movement on the suggested path by receiving the user's position coordinates from an external location sensing component. The system matches these coordinates with path elements of the navigation space, exploiting the SVG technology of the map files. Finally, the guide dynamically re-plans a user's route from the current position in case he/she has deviated from the suggested path. A very important system characteristic is the semantics-based modeling of user motion and location, a feature which provides a powerful advantage in the indoor navigation research field, as the potential for describing relations between humans and space elements is almost unlimited. Another important feature of the introduced Contextual Guide is the management of user interaction regarding navigation issues. In fact, the system is able to receive user feedback about external factors that may render the proposed path non-traversable and, with the aid of reactive rules, to dynamically re-plan the user's route, providing a new, collision-free route to the desired destination.

Σημασιολογική Παρακολούθηση Χρήστη και Δυναμικός Ανασχεδιασμός Μονοπατιού για έναν Οδηγό Πλοήγησης

Εμμανουήλ Νικολουδάκης
Μεταπτυχιακή Εργασία
Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

ΠΕΡΙΛΗΨΗ

Η ερευνητική δραστηριότητα στον τομέα της ανθρώπινης πλοήγησης τα τελευταία χρόνια έχει αυξηθεί σημαντικά. Σε ολοένα και περισσότερα παραδείγματα της καθημερινότητας η ηλεκτρονικά υποβοηθούμενη πλοήγηση εφαρμόζεται με απόλυτη επιτυχία. Επιπλέον, οι ηλεκτρονικές συσκευές πλοήγησης έχουν επηρεάσει σημαντικά την διαδικασία αναζήτησης κατάλληλου μονοπατιού και εύρεσης του τελικού προορισμού, ειδικά με την βοήθεια της τεχνολογίας GPS που χρησιμοποιείται στους εξωτερικούς χώρους. Εντούτοις, εξετάζοντας τα υλοποιημένα συστήματα πλοήγησης για εσωτερικούς χώρους, είναι φανερό ότι υπάρχει δυνατότητα για τη σχεδίαση και ανάπτυξη καινοτόμων ιδεών, ιδιαίτερα στις διαδικασίες παρακολούθησης χρήστη και διαδρασής αυτού με το σύστημα. Εκμεταλλευόμενοι αυτή τη δυνατότητα, το αρχικό κίνητρο είναι η δημιουργία ενός σημασιολογικού οδηγού (contextual guide) για εσωτερικούς χώρους, σχεδιασμένο για πλοήγηση πεζών με βάση το προφίλ τους. Ο σημασιολογικός οδηγός μπορεί να χωριστεί σε τρεις διαφορετικές ερευνητικές ενότητες. Η πρώτη ενότητα ασχολείται με τη δημιουργία και διαχείριση του προφίλ του χρήστη, ενώ η δεύτερη εστιάζει στη χωρική αναπαράσταση και τις μεθόδους εύρεσης μονοπατιού. Τέλος, η τρίτη ερευνητική ενότητα, που αντιπροσωπεύει και το θέμα της παρούσας εργασίας, ασχολείται με την παρακολούθηση του χρήστη κατά τη διάρκεια της πλοήγησης του. Συγκεκριμένα, το σύστημα είναι υπεύθυνο να παρατηρεί την διαδρομή του χρηστή στο προτεινόμενο μονοπάτι, λαμβάνοντας σαν είσοδο τις συντεταγμένες θέσης του από ένα εξωτερικό σύστημα εντοπισμού θέσης (location sensing). Το σύστημα αντιστοιχεί τις συντεταγμένες θέσης πάνω στο χώρο πλοήγησης χρησιμοποιώντας τις δυνατότητες της SVG τεχνολογίας που βασίζονται οι χωρικοί χάρτες. Σε περίπτωση που ο χρήστης αποκλίνει από την προτεινόμενη διαδρομή, ο οδηγός ανασχεδιάζει δυναμικά το μονοπάτι από την τρέχουσα θέση του χρήστη. Ένα πολύ σημαντικό χαρακτηριστικό της αρχιτεκτονικής του συστήματος είναι η σημασιολογική μοντελοποίηση της κίνησης και θέσης του χρήστη που παρέχει ισχυρό πλεονέκτημα στον τομέα της πλοήγησης σε εσωτερικούς χώρους, αφού οι δυνατότητες περιγραφής σχέσεων μεταξύ ανθρώπων και στοιχείων μονοπατιού είναι σχεδόν απεριόριστες. Τέλος, ένα ακόμα σπουδαίο στοιχείο που εισάγει ο σημασιολογικός οδηγός είναι η διαχείριση της διάδρασης του χρήστη που σχετίζεται με θέματα πλοήγησης. Στην πραγματικότητα, το σύστημα μπορεί να λαμβάνει την ανάδραση (feedback) από τον χρήστη αναφορικά με εξωγενείς παράγοντες που εμποδίζουν την ομαλή πλοήγηση του. Στη συνέχεια, με τη βοήθεια κανόνων (reactive rules) το σύστημα εκτελεί δυναμικό ανασχεδιασμό, προτείνοντας στον χρήστη ένα καινούργιο μονοπάτι χωρίς εμπόδια, προς τον επιθυμητό προορισμό.

Επόπτης Καθηγητής: Δημήτριος Πλεξουσάκης
Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Acknowledgments

The author would like to thank all the persons who provided support and encouragement throughout the pursuit of this research project. In particular my supervisor, Mr Dimitrios Plexousakis along with Mr Grigorios Antoniou, professors of Computer Science Department (University of Crete), who inspired the idea of the ‘Contextual Guide’ project and appointed me for the ‘User Tracking and Dynamic Re-Planning’ research part. Their guidance and suggestions were very helpful and a key factor for the completion of this work. Moreover, the author wants to express his gratitude to the Ph.D. candidates Theodore Patkos and Antonis Bikakis for organizing and directing his thoughts, constantly advising and suggesting the best solution for every problem. Also to the project partners, Maria Michou and Manolis Kritsotakis, master’s candidates, for the excellent cooperation, the useful answers in author’s innumerous questions and the direct response in his every need, especially during the implementation process. Finally, the author would like to acknowledge the mental support provided by Garifalia Kouzouloglou, during the research, design and implementation phases of this master’s thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General statement of the thesis subject

In the recent years, the increasing user demand for continuous web information access and the technological advances in mobile devices and applications caused a boost in the progress of wireless personal communications. This revolution facilitated the vision for pervasive services, which aid users in their everyday life activities in an intelligent and discrete way. Enriching system modeling with semantics aimed at developing applications with advanced functionality and greater reasoning capabilities. Pervasive computing environments can achieve the vision of Ambient Intelligence by combining domain knowledge with advanced reasoning mechanisms, allowing the deployed services to explore hidden relationships between the system entities and to provide solutions to problems that were much complicated.

Difficult to ignore that last years activities in the navigation field have been increasing drastically. There are and will be more and more areas where navigation becomes an important part of a system solution. So far navigation systems have been of major importance for aircraft, missiles, ships, but it is now becoming used in automobiles, mobile phones and other smaller systems.

Electronic navigation devices have strongly influenced the process of human wayfinding (pathfinding) in the last years. Vehicle navigation systems are already in widespread use and the appliance of navigation systems in other domains such as hiking, biking or exploring cities on foot is continuously increasing. Different information needs coming from various domains bring up a number of new research questions which have to be considered in order to design navigation aids for these domains.

Wayfinding is one of people daily activities. Finding a way from A to B is typically a non-trivial task which requires a number of cognitive abilities and extensive spatial knowledge. Today the process of pathfinding is more and more supported by different aids, for example cartographic maps. With the availability of global positioning systems and technological progress in the field of mobile computing, also electronic wayfinding aids have found their way in every day lives.

Continuously decreasing hardware sizes and reduced power consumption of GPS receivers has prepared the way for GPS technology into small navigation devices. These devices can be carried in the hand or in the pocket while performing outdoor activities. It's the beginning of a new age in navigation technology which could be called the age of 'ubiquitous navigation'. The term ubiquitous navigation indicates the process of personal navigation, supported by small electronic devices, which are characterised by the ability to provide navigational assistance in nearly every wayfinding situation.

With new technological opportunities navigation domains are also changing. A broad variety of potential new navigational domains opens the market for exciting new applications. But what are the requirements for electronic navigation systems in these domains? A look at current electronic navigation aids shows some of the problems. Needless to say that the majority of the navigation solutions are targeting the outdoor environments by exploiting the tracking advantages that GPS applications provide. The market is full of personal navigation systems, which are designed for assisting car drivers. The systems rely on digital road maps, which are also used for generating wayfinding instructions. However, in order to be prepared for new markets, these datasets and the corresponding navigation solutions should now be adapted to the information needs of pedestrians or hikers.

On the other side, examining the already launched indoor navigation systems, it is obvious that opportunities for innovative research suggestions are available. Taking advantage of these opportunities, the current thesis makes an effort to promote existed research and contribute to the field of indoor navigational systems. The original motivation is to create a semantics-based contextual guide for indoor environments. The contextual guide will be designed for pedestrian navigation regarding user’s physical and mental profile.

An example scenario of the contextual guide’s appliance is following. Suppose a visitor who enters an exhibition building and wants to see a specific part of the exhibition somewhere inside the building. His/her first step is to create a profile inside the guiding system and register all the appropriate information, including physical characteristic and preferences. Physical characteristics are stored for the optimal path calculation, whereas stored preferences will determine the destination location, which is, visitor's desired exhibits. In more detail, if the visitor is handicapped (physical characteristic), the pathfinding part of the system will suggest a route with motor passages (e.g. elevators). Furthermore the guiding system will present the exhibits (points of interest, POIs) with the appropriate description (textual, visual, audio) according to the profile preferences. The guiding system will also have the responsibility to track visitor's position, inform him/her about current navigation location and detect any deviations from the proposed path. In such a case, the system should be able to dynamically re-plan visitor's route, suggesting a new path from the deviated position.

Of course, the above scenario uncovers only a small portion of the full potential which the desired contextual guide application should include. The design plan to fulfil the original expectations led the research team to divide the total study into three different theme blocks. The first block is occupied with the design and implementation of dynamic semantic profiles and the development of services that use these profiles. The second theme block is focused on semantic spatial representation and the development of path finding and navigation techniques. Finally, the third theme block, is occupied with the user tracking, during navigation, and dynamic re-planning in case of deviation from the proposed path.

For the semantics profile construction (first theme block) developed by M. Michou, users are classified into user groups according to information, such as: their sex, their research interests, their occupation, their disabilities and their preferences. The second theme block, developed by M. Kritsotakis, especially handles the wayfinding process. Paths from some position A to some other position B are usually computed by a shortest path algorithm in a graph. These algorithms only need the nodes and edges of a graph, together with a cost function. It turns out that more complicated than computing a shortest path is the problem of turning the shortest path into a human understandable description. A lot of additional information is needed to generate useful path descriptions. Therefore, a hybrid location model is developed which combines semantic and geometric information to serve both, the shortest path algorithms and the generation of relations and descriptions.

The last theme block mentioned above, is the subject of current thesis. The navigation system receives input from a separate location sensing application which communicates with the user by an equipped electronic device, for example a pda. This particular input is the coordinates of the user in the indoor environment. The contextual guide represents on the pda the exact location by highlighting the coordinates and forwards to the user's device all the available information taken from the location semantics.

The related information which is projected on user's pda consists of user location descriptive details and the orientation of user's movement. For example, if user walks down a corridor X on first floor of a building, the guiding system expresses (e.g. in textual format) this situation as follows: "Visitor A is moving inside corridor X, on floor1 of building". Since the corridor has a particular orientation, the system applies this orientation to user's movement. So if corridor X is north, the navigator's feedback about user's movement will be: "Visitor A is moving north, inside corridor X, on floor1 of building".

The necessity for a better communication between system and user generated the contextual guide's ability to receive dynamic feedback from the user. For example if during his/her navigation on the proposed path an obstacle blocks the way, user reports the incident and the system is responsible to dynamically re-plan suggested path from user's current location and to propose a new valid route.

Another responsibility of the system is the continuous tracking of the user during navigation. The system should compare the suggested path with the coordinates received from location sensing application and make sure user follows the suggested path. In case of deviation, the system should detect it and use the dynamic re-planning method in order to execute the pathfinding algorithm from current user's position (as algorithm's new starting point) to the same destination. As a result the system highlights the new proposed path on user's pda to the desired destination and user tracking process starts again.

During user's movement on the path, the navigator represents on pda's map the points of interest (e.g. exhibits) which exist inside user's nearby area. Moreover, the system forwards the stored description for the particular points of interest and informs user for POI's relative location (e.g. POI is on visitor's left side). Using this particular service, user can express his/her preferences via the profile manager and, as a result, the system will have the ability to represent on the map only the user's desired POIs. This feature will definitely save from the user precious time and he/she will avoid the tiring experience of examining all the POIs on the area.

1.2 Definition of terms

In this section, in order to make the reader acquainted with the basic terminology, some specific terms which have been mentioned above, will be described in more detail.

1.2.1 Ontologies

Ontologies and semantic representation typically represent scientific and research field of knowledge representation and artificial intelligence, in general. The term semantics means all the meta-data that are related with an object or a model and help in comprehension of the characteristics and elements, such as constraints, causes, relations with other models or objects. Semantic representation is of great importance for the developing systems, services and applications, because gives the ability for better understanding of the ideas involved and the conduction of useful conclusions. In computer science, for simplicity reasons, semantics is all the meta-data which shape full descriptions for the systems components.

The increasing need for the expression of semantics led the knowledge representation field in the search of methods for the formal representation of semantics. From early 90's, one particular method appeared to provide all the proper characteristics for the semantics representation. This 'method' is ontology. One of the most accurate and inclusive definition of ontology is the following:

"Ontology is a formal, explicit specification of a shared conceptualization"

(R. Studer 1998, original definition from T. Gruber in 1993)

Every word in this definition has its own meaning. First of all, the term "conceptualization" means an abstract model of a phenomenon. The description of this model is according to the specification of the basic principles, relations, constraints and axioms of the model. Moreover, this specification must be expressed in formal format in order to be machine readable. Finally, the conceptual of this phenomenon should be common for all related parties.

In a technical level, an ontology is a class hierarchy and the more its soundness and formalism is expanded, the more elements are added in this initial hierarchy. These elements could be the attributes, the restrictions, the axioms and the relation or roles. In general, an ontology is not far from the sets of classes that exist in the object oriented programming or Entity-Relationship models in database architecture. The main difference is that in every designed element is attributed an explicit formal semantics. So an ontology is described with the semantic help of its description language.

Ontologies are often equated with taxonomic hierarchies of classes, class definitions and the subsumption relation, but ontologies need not be limited to these forms. Ontologies are also not limited to conservative definitions, such as definitions in the traditional logic sense that only introduce terminology and do not add any knowledge about the world (Enderton, 1972).

To sum up, the semantics of a term is a set of meta-data and the most appropriate method of its representation is by using ontologies. When an ontology is designed and created for a specific domain of interest, the main goal is to preserve a generally accepted vocabulary and to exploit all the benefits of conducting conclusions using the related semantic information.

1.2.2 Pervasive Computing

Pervasive computing is a rapidly developing area of Information and Communications Technology. The growing availability of microprocessors with inbuilt communications facilities, made possible for pervasive computing to enter into people's lives and environments. Pervasive computing has many potential applications, from health and home care to environmental monitoring and intelligent transport systems.

The goal of pervasive computing, which combines current network technologies with wireless computing, voice recognition, Internet capability and artificial intelligence, is to create an environment where the connectivity of devices is embedded in such a way that the connectivity is discreet and always available. Pervasive computing devices are not personal computers as someone could think of them, but very tiny, even invisible, devices, either mobile or embedded in almost any type of object imaginable, including cars, tools, appliances, clothing and various consumer goods, all communicating through increasingly interconnected networks.

The idea of Pervasive or Ubiquitous Computing was firstly expressed by Mark Weiser in 1991 [6]. This idea describes a new model of computer environments in which there exist many microcomputers that cooperate to provide to the user a total new experience. The integration of such a model includes several stages and different research fields. Some of them are:

- Smart adaptive user interfaces.
- Cyberforaging
- Proactivity
- Smart Spaces

One of the fundamental principles of pervasive computing is the human-centered system design, both hardware and software. User should interact and exploit system functionality without even realizing the existence of the system devices or without knowing system's technical characteristics. The services that are suggested by this model are personalized for every user and take into consideration the contextual information, such as user profile, position, environmental conditions and security restrictions, in order for the model to fulfill the user's needs successfully. Obviously, these needs should not be defined explicitly from the user, but the system should have the ability to conduct them from the contextual information. The final result of such a confrontation of user needs, leads to a human-centered system where user is neither disturbed nor annoyed by system behavior.

There is a wide range of potential benefits for government, service providers and consumers as computing technologies become more pervasive. There is debate over how to address concerns over privacy, security, safety and sustainability while still realising the benefits of pervasive computing. Such concerns may need to be addressed by means of voluntary guidelines, legislative measures, physical design, or a combination of these.

1.2.3 Semantic Web

The value and the contribution of the World Wide Web (WWW) in the relationship between human and computer is obviously great. WWW gave the ability to millions of people to communicate and interact, no matter where they are, in which ethnicity they belong and what are their interests. Nowadays, a very large part of human communication via computers is covered by internet and its basic application, WWW.

However, WWW faces a serious weakness; it is based on the syntactic description of the content, in order to be comprehended by humans. As a result web pages only contain information in human understandable format, such as texts, pictures and videos and make it impossible for machines to automatically process internet data. Such a data process presupposes a proper description of the published content and algorithm that will add the necessary intelligence in the machines. Such a description should contain, apart from the basic human-readable content, data that will be purposed exclusively for computer processing and will include content semantics.

The above weakness of WWW and the proposed solution was visualized by Tim Berners-Lee [3], who named the evolution stage of WWW as Semantic Web.The Semantic Web targets on structuring the content of web pages and creating an environment where software agents traversing the

WWW would have the ability to execute advanced operations for users. The Semantic Web is not a different internet but an expansion of the current WWW, in which data are characterized from structured content, giving the possibility for an advanced communication between human and computer, since there will be a common language between them, the semantic description.

Semantic web is the base of a complete distributed form of artificial intelligence. As it is well known, artificial intelligence is basically occupied with two subjects: knowledge representation and methods of searching and reasoning. In order for the Semantic Web to function properly, computers should have access in structured knowledge bases and in inference rules which could be used for executing automatic reasoning. The existed technologies of knowledge representation have made some progress but they are still not able to fulfill the expectations of artificial intelligence. The traditional methods of knowledge representation presuppose a common, universal knowledge base and rules so as everybody use them. On the other hand, the Semantic Web does not impose the absolute conformity among the systems which are interconnected, in accordance with the convention of WWW: there is no certainty that you will find something you are seeking, even if it exists. Despite the cost, however, it provides flexibility and expressiveness since it does not limit the process of knowledge representation. Thus a challenge for Semantic Web is to supply with a language which can express data and rules of reasoning and will permit the proper functionality of every existed system of knowledge representation in WWW.

The Semantic Web is not a vision for the future of the Web any more, it is becoming a reality , in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web [7]. The Semantic Web will build on XML's ability to define customized tagging schemes and RDF's flexible approach to representing data. The first level above RDF required for the Semantic Web is an ontology language that can formally describe the meaning of terminology used in Web documents. If machines are expected to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema.

1.2.4 Location Based Models

Location information is crucial for many mobile and Pervasive Computing applications. Location is used for a variety of purposes, e.g., to track people, to guide visitors, to trigger events or to route communication packets. These systems use a location model to represent different locations. The model allows one to distinguish between different locations, to compute with them, for example to compare locations or calculate distances, or to present the information to the user. The choice of a location model has implications on the usability of the applications as well as on the ease of implementation. Furthermore, different application domains might need different types of location models. Below, two categories of location modeling are presented:

- Geometric models: The first category is based on Euclidean geometry and defines location by way of coordinates, usually chosen as orthogonal (a.k.a. Cartesian) coordinates relative to a given Coordinate Reference System. Technologies such as scene analysis will usually provide their output according to such a model.
- Semantic models: In this model, location concepts are defined relatively to a given universe of discourse such as architecture, physical geography, political geography, city planning, etc. The perspective is to link a mathematical definition of position (as in the geometric, the set or the structural model) to a more human friendly notion of place as in [11]. The goal is to automate the process with an explicit definition of the semantics for the computers to understand it in a ubiquitous environment.

In order to access the situational context of the user within the intelligent environment, the gap between the coordinate based pedestrian positioning system and the symbolic ubiquitous world mode must be bridged. There are good reasons to use both models:

- To express the user's situational context, symbols to denote locations are needed.
- Geographical Coordinates are not useful to represent locations in multi-level indoor environments.
- In order to compute the user's location from numeric sensor information (e.g. GPS) and to provide the user with situated navigational aid ("go 5 meter, then turn left") geometrical knowledge is needed.
- For advanced visualization, e.g. from an egocentric perspective, a true 3D model is required.

A hybrid location model is a well-defined structure of data about a situation according to its location. A hybrid location model can make the description of a physical situation exact and rich: it has both geometrical and semantic information. Geometrical information depends on distance measurements, while semantic information refers to relationships among spaces, spatial elements, objects and users.

A location-aware system must be based on a well-defined location model. It is recognised from all above that the chosen location model determines the functionality supported by a location service. It became evident that the location model also has major architectural implications. A simple yet powerful model, which is free of immediate dependencies on sensor technologies or application domains, should provide the key to solving issues of security, scalability and manageability.

1.2.5 Context-Aware Systems

Context is a powerful, and longstanding, concept in human-computer interaction. Context can be used to interpret explicit acts, making communication much more efficient. Thus, by carefully embedding computing into the context of human activities, it can serve with minimal effort on user part. Communication can be not only effortless, but also naturally fit in with all user ongoing activities. Pushing this further, the actions taken are not even felt to be communication acts at all; the user is rather just engaged in normal activities and the computation becomes invisible.

One challenge of mobile distributed computing is to exploit the changing environment with a new class of applications that are aware of the context in which they are run. Such context-aware software adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time. A system with these capabilities can examine the computing environment and react to changes in the environment [26]. Three important aspects of context are: where you are, who you are with, and what resources are nearby. Context encompasses more than just the user's location, because other things of interest are also mobile and changing. Context includes lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation, for example whether user is with his/her manager or with a co-worker [15].

One goal of context-aware computing is to acquire and utilize information about the context of a device to provide services that are appropriate to the particular people, place, time, events, etc. For example, a cell phone will always vibrate and never beep in a concert, if the system can know the location of the cell phone and the concert schedule. However, this is more than simply a question of gathering more and more contextual information about complex situations. More information is not necessarily more helpful. Further, gathering information about user activities intrudes on his/her privacy.

1.3 Description of the remaining chapters

This first chapter introduced the subject of the master's thesis and presented briefly the main points of research and integration which will be described in later chapters in detail. Furthermore, a separated section was dedicated to inform the reader about specific terminology, in order to get familiar with useful definitions and background knowledge, essential for the comprehension of current thesis.

Chapter 2 will be an effort to present and summarize the main characteristics of some important integrated navigation systems and to mention not only their positive points but their weaknesses as well. Then, a small review of the literature related to location-based modeling systems will be delivered and finally a small section about the innovative themes, this particular thesis introduces, will be outlined.

Chapter 3 will state the full system design and architecture, while chapter 4 will describe all the implementation techniques as well as the system information flow. In chapter 5 all the design and implementation tools, which have been used, will be mentioned there. Finally, in chapter 6, a brief summary and all the thesis concluding ideas will be stated, as well as the opportunities for future work.

Chapter 2
Literature Review

2.1 Integrated Navigation Systems

As mentioned in the previous section, then general goal of the thesis is the creation of an indoor contextual navigator that will have the ability to guide pedestrians through buildings. This section is an effort to describe the main navigation systems that have already been integrated and the fundamental related research topics that other authors have developed.

2.1.1 Cyberguide

The research and development for navigation systems has been improved the recent years. One of the first systems oriented for human guiding was Cyberguide, which served both indoor and outdoor navigating purposes. It was firstly designed as a tool to aid tourists inside museums and archeological sites according to the position of the user and his/her direction [2]. The challenge addressed in Cyberguide is how to build mobile applications that usefully leverage off information about the context of the user. Cyberguide provides a position-aware handheld tour guide for directing visitors around the GVU Lab (GVU Center & College of Computing, Georgia Institute of Technology).

Visitors to a GVU open house are typically given a map of the various labs and an information packet describing all of the projects that are being demonstrated at various sites. Collapsing all of the paper-based information into a handheld intelligent tour guide that knew user's location, his/her intentions and could answer typical questions provided a test bench for research questions on mobile, context-aware application development. The architecture of Cyberguide consisted of four independent components: the map, the information base, the positioning system and the communications system.

The map is the view which visitor is using to navigate. Visualizing and manipulating the map dominates the user interface of Cyberguide. It can be viewed at varying levels of detail and scrolled around. The visitor is indicated by location and orientation on the map and various demonstrations are also marked. Information on a demonstration is revealed by an explicit pen touch on the map or by wandering "close" to a demo. Touching the name of the demo will move the user in hypertext fashion to an information space component that describes relevant information on the project and people associated with that particular demonstration. The positioning component provides constantly updated information on the location and orientation of the tourist. To facilitate communication, an application-level protocol on top of Appletalk was designed. This communication mechanism permits a user to send e-mail, print documents, and eventually communicate with other Cyberguide users.

2.1.2 Navio

A later and more improved solution which took into consideration the human factor and contained human-centric criteria was Navio system [14]. Navio, stands for Pedestrian Navigation Systems in Combined Indoor/Outdoor Environments, targeted in the development of an ontology that would provide guiding information to the users describing the criteria, the actions and the reference objects that pedestrians use during their routing. Project mainly focused on the information aspect of location-based services, especially on the user’s tasks and the support of the user’s decisions by information provided.

Specifications allowed to select appropriate sensor data and to integrate data when and where needed, to propose context-dependent routes fitting to partly conflicting interests and goals as well as to select appropriate communication methods in terms of supporting the user guidance by various multimedia cartography forms. These tasks are addressed in the project in three different work packages, the first on “Integrated positioning”, the second on “Pedestrian route modeling” and the third on “Multimedia route communication”. However, Navio emphasises on location fusion and user interface and does not contribute on the subject of the path-finding.

In the work package “Integrated positioning” of the research project Navio, which is the closest to the research area of current thesis, the following challenging tasks are addressed:

- The capability to track the movements of a pedestrian in real-time using different suitable location sensors and to obtain an optimal estimate of the current user's position.
- The possibility to locate the user in 3 dimensions with high precision (that includes the ability to determine the correct floor of the user in a multi-storey building).
- The capability to achieve a smooth transition for continuous positioning determination between indoor and outdoor areas.

Thereby a navigation support must be able to provide location, orientation and movement of the user as well as related geographic information matching well with the real world situation experienced by pedestrians. For the indoor positioning related to the current thesis, the guidance of a pedestrian in 3-D space and updating of his/her route, continuous position determination is required with positioning accuracies on the few meter level or even higher, especially for navigation in multi-storey buildings in vertical dimension as the user must be located on the correct floor. The specialized research hypothesis of this work package in the project Navio is that a mathematical model for integrated positioning can be developed that provides the user with a continuous navigation support. Therefore appropriate location sensors have to be combined and integrated using a new multi-sensor fusion model.

Navio, as it is obvious from all above, strongly emphasises on location fusion and user interface, but its contribution on the pathfinding section and especially on the correspondence of an appropriate path for the specific needs of every user, is relatively poor.

2.1.3 myCampus

Carnegie Mellon University integrated after a five year research effort; myCampus system, a Semantic Web environment for context-aware mobile services aimed at enhancing everyday campus life [18]. The project has drawn on multiple areas of expertise, combining the development of an open Semantic Web infrastructure for context-aware service provisioning with an emphasis on issues of privacy and usability. Work in myCampus combines technology development such as context-aware agents and location tracking functionality with evaluation methodologies. A central element of the myCampus architecture is its use of Semantic e-Wallets aimed at reconciling user demands for context awareness and privacy as well as a description of different context-aware applications developed and evaluated during the course of the project.

In myCampus, users can acquire to different sets of task-specific agents that help them with different tasks. To properly operate these agents require knowledge of one or more contextual attributes about their users as well as possibly other users. These attributes can potentially be acquired from a number of possible resources, which typically vary from one user to another and may even vary over time for the same user. To overcome this challenge, sources of contextual information in myCampus are modeled as Semantic Web Services that can automatically be discovered and accessed by agents. Access to a user's contextual resource is controlled according to user-specified privacy preferences.

The e-Wallet Manager serves as a repository of static knowledge about the user, except that here knowledge is represented using OWL. In addition, the e-Wallet contains knowledge about how to access more information about the user by invoking a variety of resources, each represented as a Web Service. This knowledge is stored in the form of rules that map different contextual attributes onto one or more possible service invocations, enabling the e-Wallet to automatically identify and activate the most relevant resources in response to queries about the user's context. User-specified privacy rules, also stored in the e-Wallet, ensure that information about the user is only disclosed to authorized parties, taking into account the context of the query. Clearly, agents are not limited to accessing information about users in the environment. Instead, they also typically access public Web.

The MyCampus infrastructure has been instantiated in the context of several prototype Ambient Intelligence environments, including:

- An environment aimed at enhancing everyday campus life at Carnegie Mellon University.
- A museum tour guide environment developed for the National Museum of Natural Science in Taiwan.
- A smart office environment.

The MyCampus environment and its applications have been evaluated through a series of experiments in which users were observed and data collected over periods of several days at a time. The priority of the system to the user's privacy and preferences is obvious from all mentioned above; however, there is not extended development on the field of the human-centric navigation.

2.1.4 OntoNav

A science group of Department of Informatics & Telecommunications, University of Athens, developed Ontonav, an integrated navigation system for indoor environments which uses a hybrid modeling, both geometric and symbolic [21]. OntoNav is purely user-centric in the sense that both the navigation paths and the guidelines that describe them are provided to the users depending on their physical and perceptual capabilities as well as their particular routing preferences. Physical capabilities include the user's capability to walk, to see, to use stairs, whereas perceptual capabilities mean how easily one can be guided in an unknown environment. These latter capabilities depend usually on the user's age and/or cognitive status. Routing preferences include user defined points of interest that should be included to the identified path, and preferences that rely on the semantic attributes of the path. OntoNav is comprised of the following building blocks:

- Navigation service: This service can be defined as the interface between the system and its users. It accepts navigation requests and responds with the optimal path, if any.
- Indoor navigation ontology: This spatial ontology, named Indoor Navigation Ontology (INO), describes the basic spatial and structural concepts of indoor environments, as well as their relationships. Specifically, it provides a semantic spatial model for reasoning about the selected paths.
- User navigation ontology: In order to model user context, a specific ontology has been developed, named User Navigation Ontology (UNO). This ontology contains user classes and elements of the user context. Each user profile is classified into one or more navigation classes according to its characteristics. Both the INO and UNO ontologies have been modeled through OWL.
- Path selection rules: The path selection process is performed through a set of rules. The definition of such rules also involves the spatial semantics and the user semantics. The rules are applied to the INO instances in order to determine the paths that are considered appropriate and accessible for each user request.
- Indoor geospatial model: The INO instances are created through a geometric representation of the indoor topology. Such geometric data may initially reside in a Geographic Information System (GIS) as building blueprints and be transformed to actual spatial ontology instances.
- Routing algorithm: This algorithm is a central element of the framework and, in combination with the Path Selection Rules, is responsible for the determination of the optimal path between two given endpoints. The algorithm used in OntoNav is a k-shortest paths searching algorithm. The main idea is that the shortest path may not always be the optimal path.
- Indoor positioning system: OntoNav symbolically locates the users in the navigation space according to the spatial model described by INO. It is

important to note that, in cases where the positioning accuracy is better than the location modelling granularity, an approximation error may be introduced in the estimated location of the user. However, this error does not significantly affect the quality of navigation.

As mentioned above, the main objective of the system is to provide a user-centric navigation paradigm for indoor environments based on the user's physical and perceptual capabilities and limitations. In order to achieve this objective, the system is aware of the user capabilities, which are described by the user profile (UP). A UP is defined as a collection of classified attributes, most of which represent specific user capabilities/limitations. OntoNav uses the user profiles in conjunction with various user-independent rules in order to infer which of the walkable paths are suitable for a given user and how the navigation guidelines should be presented. These two selection processes are implemented with the aid of three kinds of navigation rules, the physical navigation rules, the perceptual navigation rules, and the navigation preferences.

The proposed navigation scheme is largely based on semantic descriptions of the constituent elements of navigation paths, which, in turn, enable reasoning functionality. Thus, Ontonav project group developed the Indoor Navigation Ontology (INO), so as to suit both the path searching and the presentation tasks of a navigation system. The INO ontology is the basic skeleton on which M. Kritsotakis developed his navigation ontology, the one used in thesis demonstration application.

2.2 Ontology Based Location Modeling for Navigation Systems

Location based modeling is a problem that occupies scientists for many decades. Many models have been proposed, but most of the scientists agree that there are two main modeling categories, symbolic and geometric. The first is based on the description of the space according to the name, the functionality, the attributes and characteristics. The second is based on the geometrical attributes of the space, such as the coordinates and the shape. Each of the two modeling categories improves different service and spatial querying. For example a symbolic model is not able to answer queries related to distance, nearest neighbors or shortest paths. So a hybrid modeling, both symbolic and geometric, could combine the advantages of the two distinct modeling categories.

A very informative book (workshop proceedings) about location modeling is “Location Modeling for Ubiquitous Computing” [4], which aims to develop an understanding of how to model location information, including the following topics:

- Enumerate and compare existing location sensing technologies and their underlying location models.
- Consider the usefulness of existing location.
- Present, assess and compare new models.
- Assess how models and data might be shared for reuse.
- Explore the complementarities and potential synergies between different location models.
- Understand how technical parameters should be represented in a location model.

Existing approaches for indoor navigation were driven by geometric information and neglected important aspects like the semantics of points/areas and user preferences. The derived applications were not intelligent enough to contribute to the pervasive computing vision. In this study [19], a novel navigation mechanism is introduced. Such a navigation scheme is enriched with user profiles and the adoption of an ontological framework. Another study [13] describes COBRA-ONT, an ontology for supporting pervasive context-aware systems. COBRA-ONT, expressed in the Web Ontology Language OWL, is a collection of ontologies for describing places, agents, events and their associated properties in an intelligent meeting room domain. Finally, a very interesting research is presented in [20], suggesting a user model on a Semantic Web ontology for navigation systems (mainly pedestrian), which is based on relevant human wayfinding and navigation theories.

One of the most important researches in the field of location modeling was made by Haibo Hu and Dik-Lun Lee, Semantic Location Modeling for Location Navigation in Mobile Environment [12], and this particular knowledge was fundamental for the design of the thesis location model. The approach of the research has as main targets the automatic creation of the location model directly from ground plans and the integration of navigational and path finding services. The proposed model and the related algorithms are based on two key terms: location and exit. The term ‘exit’ describes every border of a space that permits the entrance or the exit from this space.

The main structures which are described in the particular research are two hierarchies: a location and an exit hierarchy. Location hierarchy is used to model the topological relations between spaces. In other words, every space has an appropriate place in the hierarchy based on its connectivity with other spaces. The exit hierarchy represents the distance between the spaces. The distances can not only be Euclidean but semantic as well, given they satisfy specific rules described in the research. Both hierarchies can be automatically created using the proper algorithms. These hierarchies constitute the symbolic part of the model. The geometric part is described by attributes in elements of the location hierarchy (e.g. a space is characterized by its coordination or/and its area). The basic idea is to take advantage of the geometric information in order to create the hierarchies and then execute the algorithms according to the created hierarchies.

Furthermore, the research paper illustrates the whole modeling process through an example and shows how cartographic information on locations and exits, typically found in ground plans and maps, can be modeled into location and exit hierarchies, from which spatial semantics, such as the topology relationships and distance between locations, can be derived and stored. In addition, this model supports end-user queries and operations, such as shortest path queries and nearest neighbour search.

Three more location modeling systems will be mentioned below. The first one, purely symbolic model, named Vitruvius, can authorize and store locations efficiently [23]. The location database can be used effectively as predefined communication interface information in the ubiquitous computing system. Vitruvius has the ability to semantically recognize human location related with the space and spatial elements and to indicate the location based on the symbolic location model.

The second, which is dedicated on the creation of a hybrid context ontology and a location-based context model, is presented here [24]. The designed ontology is called COMANTO (COntext Management oNTOlogy) and describes general context types and interrelationships that are not domain, application or situation specific. The location-based context model proposed focuses on addressing context management challenges in distributed pervasive environments and is integrated with the COMANTO context knowledge. The combined modeling approach aims to enable efficient management of context data and allow for a widely applicable context formalism.

Finally, a system using location and motion modeling is used here [25]. One of the main purposes of the walk ontology proposed is to capture navigation instructions in a language and walking direction independent way. This means that an author is able to create navigation instructions by using his/her language and by describing one walking direction. The system should then be able to reuse that information for generating instructions in other languages and/or the other walking directions. For this reason authors should not start with typing navigation instructions manually but they will use a limited vocabulary. Therefore they will select from a list of possible, standardised instructions and other building blocks to enter additional information. Three parameters characterize each individual navigation instruction: the type of navigation instruction, the direction in which the hiker should proceed and the combination of a geographic feature and a spatial relation.

2.3 The innovative ideas introduced

In the two previous subsections, while studying the background information of the thesis, an effort was made to present the basic structure of some important navigation systems. It was an effort to highlight their positive suggestions in the research field as well as to uncover their weaknesses. Some of them were designed using geometric location modeling and as a result their contribution to the semantically enriched contextual guiding was poor. Others were designed according to the semantic location model, so the path finding part of the application was weakened. Finally the systems which introduced the hybrid location model were mostly dedicated on the user navigation services, such as the privacy, preferences and the suggestion of the optimal path. But independently of the location model, almost all the systems fairly neglected the part of user tracking.

The current thesis is an attempt to support the lack of research in the field of user tracking, during user’s navigation, as well as the modeling of both his/her location and motion. The innovative part of the thesis is the ability of the proposed system to track user’s movement after suggesting the optimal path, to check if user follows it or to detect path deviation. If users complete successfully their navigation, the system notifies them. In different occasion, the system alerts for deviation from the proposed path, and highlights a new optimal path from the location of the deviation to the desired destination. In fact the system activates a dynamic re-planning of the path finding algorithm.

Apart from user tracking and dynamic re-planning, the system has the ability to provide semantic and geographic information about the position of user in space and the direction of his/her motion. Furthermore, it can present all the points of interest in the nearby area, filtering them using user’s profile. The existence of both semantic and geographic information suggests a hybrid location model, in which the two different models are harmonically combined.

The introduction of an ontology that joins the user profile ontology and the space ontology, while expands them with a motion and location modeling ontology, also constitutes an innovative plan. The new proposed ontology has the ability to store geographic and semantic information about the space, the user location and movement, so as to make tracking an easy and secure operation. Above all, the ontology’s ability to create 3-ary relations between users, descriptions and objects, for example: “John is inside elevatorX “, is the most significant point that empowers the location modeling.

Chapter 3

System Design & Architecture

3.1 General Architecture

As mentioned in the introductory section, the contextual guide's research project is divided in three main theme chapters. The first one is occupied with the design and implementation of dynamic semantic profiles and the development of services that use these profiles. The second one is focused on spatial modeling of indoor environments and the development of path finding and navigation techniques. The third one, finally, is occupied with the user tracking process during navigation and dynamic re-planning in case of deviation from the proposed path.

Figure 3.1 shows a very general diagram of the contextual guide, its three main components and the information flow between user, guide and location sensing system. As depicted in figure below, user and contextual guide have bilinear information flow, as user has the ability to sent data to the system (e.g. by reporting a locked door) and the guiding system gives feedback to the user (e.g. by dynamically re-planning the proposed path). Furthermore, the location sensing system receives information from user's device and forwards it to the contextual guide, in the form of coordinates. In contextual guide's diagram, the user tracking component is highlighted as it is the thesis research field.

The communication between the three components is also an interesting matter. The first important relationship exists in the level of ontologies. Every component has its own ontology and a specific architecture has been adopted. Both user profile and spatial modeling ontologies import one another in order to create function rules and conditions. User tracking ontology imports the two ontologies mentioned above because of two important reasons. Firstly, user tracking process demands the access to user's profile information. Secondly, user's position and motion modeling requires direct communication with the spatial model.

The second very important relationship between the components is the need of the tracking process to communicate with the path finding algorithm. When, for example, the tracking system detects a user deviation, it should dynamically re-plan the proposed route and present the new optimal path. In this case, tracking system calls the path finding algorithm with parameters the user’s current position, as start point, and the same destination as before.

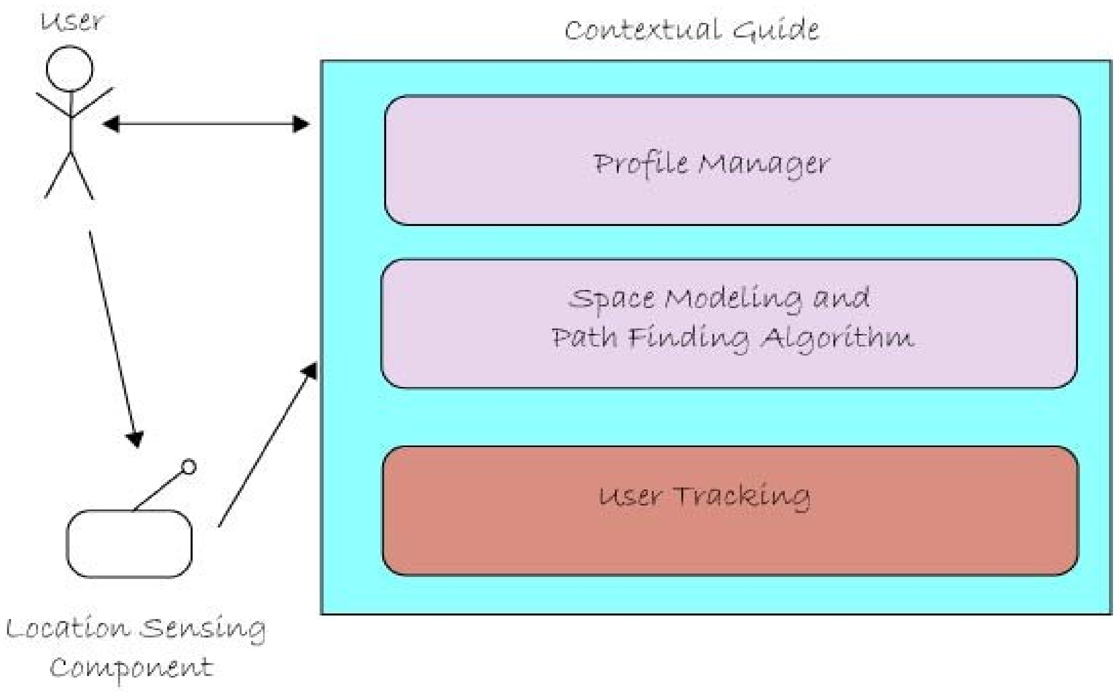

User
Contextual Guide
Profile Manager
Space Modeling and Path Finding Algorithm
User Tracking
Location Sensing Component

Figure 3.1: General architecture and information flow of the Contextual Guide

Below are briefly summarised all the case in which the user tracking system communicates with the other components:

- *User instantiation*: In the beginning of the contextual guide’s operation, the tracking system receives a user instance from the user profile ontology.
- *User location & motion modeling*: The tracking system exploits the facilities of the space modeling component to express relations between user and path elements or POIs.
- *User feedback*: User is able to give feedback in three different occasions, a locked door, a path obstacle or a malfunctioning motor passage. In these three occasions tracking system communicates with the path finding algorithm for the dynamic re-planning.

- *User deviation*: If user deviates form proposed route, the tracking system calls, in this case too, the path finding algorithm and dynamically re-plans the proposed path.
- *POIs representation*: When user traverses specific space or path elements (e.g. rooms, corridors), the tracking system checks, with the aid of space modeling ontology, if there are any points of interest in this particular area. If there exist any, the system highlights them and forwards their description stored in the ontology.

In Figure 3.2, a general diagram of the user tracking system architecture is presented. The system is separated in three distinctive layers, the ontology layer, the application program interface (API) along with the design rules and the graphical user interface (GUI). The arrows express the flow of information between those layers.

The first layer, the user tracking, motion & location modeling ontology will be analyzed in depth in section 4. The whole model of the tracking system is stored in the ontology and the functionality provided is stated below:

- *User motion modeling*: The system recognises the type and orientation of user's movement and categorizes it inside the ontology, in the appropriate class.
- *User position description*: The system expresses static and dynamic relations between user and path elements or POIs during his/her navigation, according to the proposed location descriptions of the ontology.
- *User profile information*: The system updates user profile according to three criterions, whether user follows or not the proposed path, whether he/she often or rarely deviates and whether he/she has activated assisted (user tracking process enabled) or autonomous (user tracking system disabled) navigation.

The communication between the ontology layer and the API & rules layer is expressed with the aid of Protégé-OWL API. The Protégé-OWL API is an open-source Java library for the Web Ontology Language (OWL) and RDF(S), which provides classes and methods to handle OWL files. More details about Protégé-OWL API in chapter V.

The second layer contains the application's middleware API, which communicates with both the ontology and the GUI. In fact, it forwards the model's information to the user interface and correspondingly updates the ontology according to user's behaviour and actions. Very important is also its relation with the designed rules that facilitate control process. The system's rules are divided in two categories, the proactive and the reactive rules. Proactive rules are developed in SWRL rule language and communicate both with the API and the ontology. A special package in the API has been created to handle the reactive ruled which have been developed in Jess definition rule language. Both rule categories will be described in section 6. Application's API is developed in Java object-oriented language and will be described in more detail in chapter IV.

The last layer represents the user interface and includes three different components:

- *User Interface*: Is referred to the html and JavaScript based features which are developed to present to every user the contextual guide application.
- *SVG based map modeling*: Is referred to the SVG map files which include the geographical representation of desired space and contain useful information, such as path element coordinates.
- *The GUI API*: Is referred to the JavaScript API which is developed to receive spatial information from the SVG, handle user actions and establish communication with the application's middleware API.

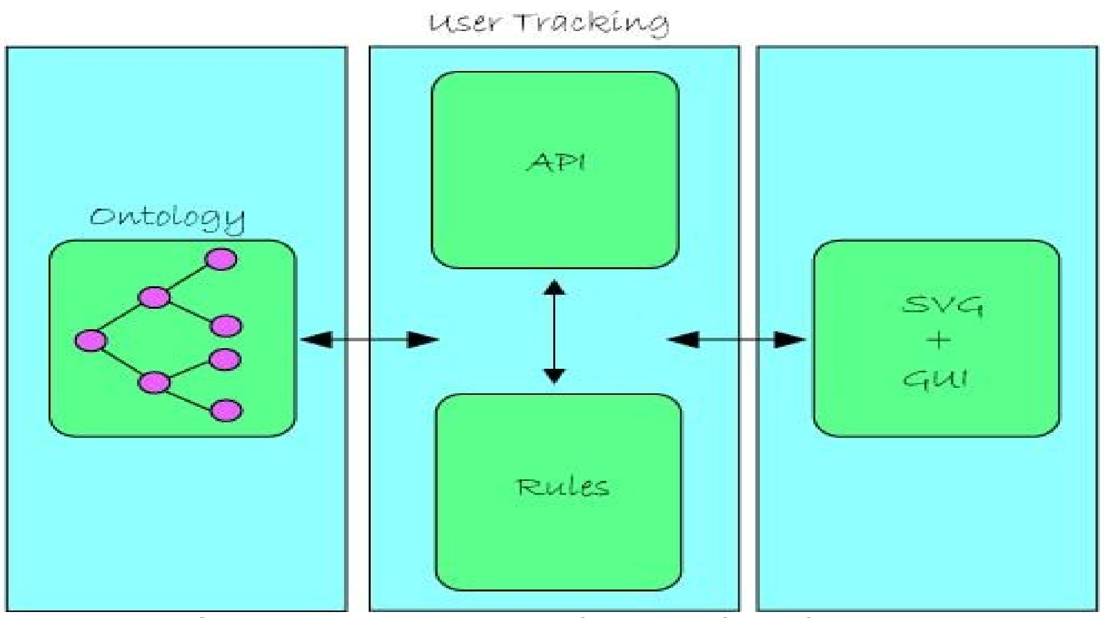


Figure 3.2: General architecture and information flow of the User Tracking component

3.2 User Profile Ontology

For the implementation of dynamic semantic profiles and the development of services which will use these profiles, a particular ontology was designed by M. Michou, part of which is depicted in Figure 3. The basic classes that are represented in the diagram are:

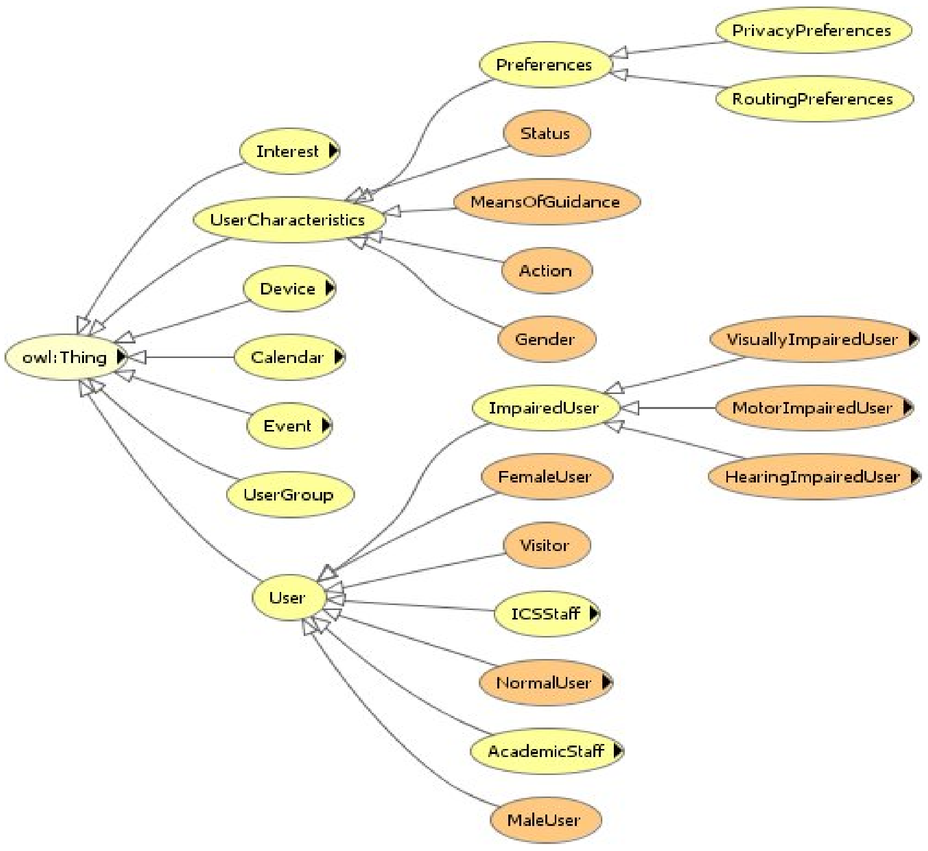


Figure 1.3: Diagram of User Profile Ontology

- *User*: The general class which models the users of the Contextual Guide, which can be divided in: impaired, female, male, visitors, ICS staff, normal and academic users.
- *UserCharacteristics*: The class which models the user characteristics stored in the ontology, which can be divided in: privacy and routing preferences, means of guidance, status, action and gender.
- *Device*: The class which models the devices that the user is equipped or may interact with.
- *Event*: The class which models all the events that take place inside the Contextual Guide's modeled space.

- *Calendar*: The class which models the private calendar of every user and stores various user data and activities.
- *UserGroup*: The class which models the groups that the users belong. These groups are created either statically (default user groups) or dynamically.
- *Interest*: The class which models the specific interests of every user (e.g. user's research interest).

3.3 Space Description Ontology

For the spatial modeling and description, a particular ontology was designed by M. Kritsotakis, which is depicted in Figure 3.4. Space Description Ontology, is based on Indoor Navigation Ontology (INO) of OntoNav project Table 3.1 shows the most important classes; followed by a description and the properties each class supports (Object and Datatype Properties).

Table 3.1: Classes, Properties and Descriptions of the Space Description Ontology

Classes	**DatatypeProperty**	**ObjectProperty**	**Description**
Path_Element		isObstacleFor hasPreferentialBonusFor hasPerceptualBonusFor hasPreferentialPenaltyFor hasPerceptualPenaltyFor	The main concept of a path.
Passage	hasLength	hasPoint	A path element that has a specific width and length. Each passage has exactly two path points
Horizontal_Passage		inFloor	A passage with all path points in the same floor

Corridor_Segment			A segment of a corridor.
Virtual_Corridor_Segment			The same as the Corridor_Segment, with the exception of belonging to a Virtual_Corridor
Door			A door that leads to some other Passage
Vertical_Passage		upperFloor lowerFloor	A passage with path points at different floors
Elevator_Segment			A segment of an elevator leading from a floor to the directly upper floor.
Elevator			The concept of elevator
Escalator			The concept of escalator
Ramp	hasGradient		A ramp that leads to another floor
Stairway			A stairway leading to another floor
Motor_Passage			A passage that operates with motor power.
Path_Point		leads_directly leads connects	A point on a path.
Exit		belongs	A point leading into or out of a space element
Navigational_Point			Points that have a particular meaning for the user navigation
Junction			A crossway of more

			than two horizontal passages
Entry_Point			A point which is a start of a virtual corridor
Space			General concept of indoors space
Corridor			General concept of a physical corridor
Virtual_Corridor			This corridor does not have a physical existence, but visualize the navigation to the interior of a space
Floor		directlyHigherFloor directlyLowerFloor	The concept of floor
Room			The concept of room
Building			The concept of building
Description			Description of a space element
Point_of_Interest			A point of interest depending on the application

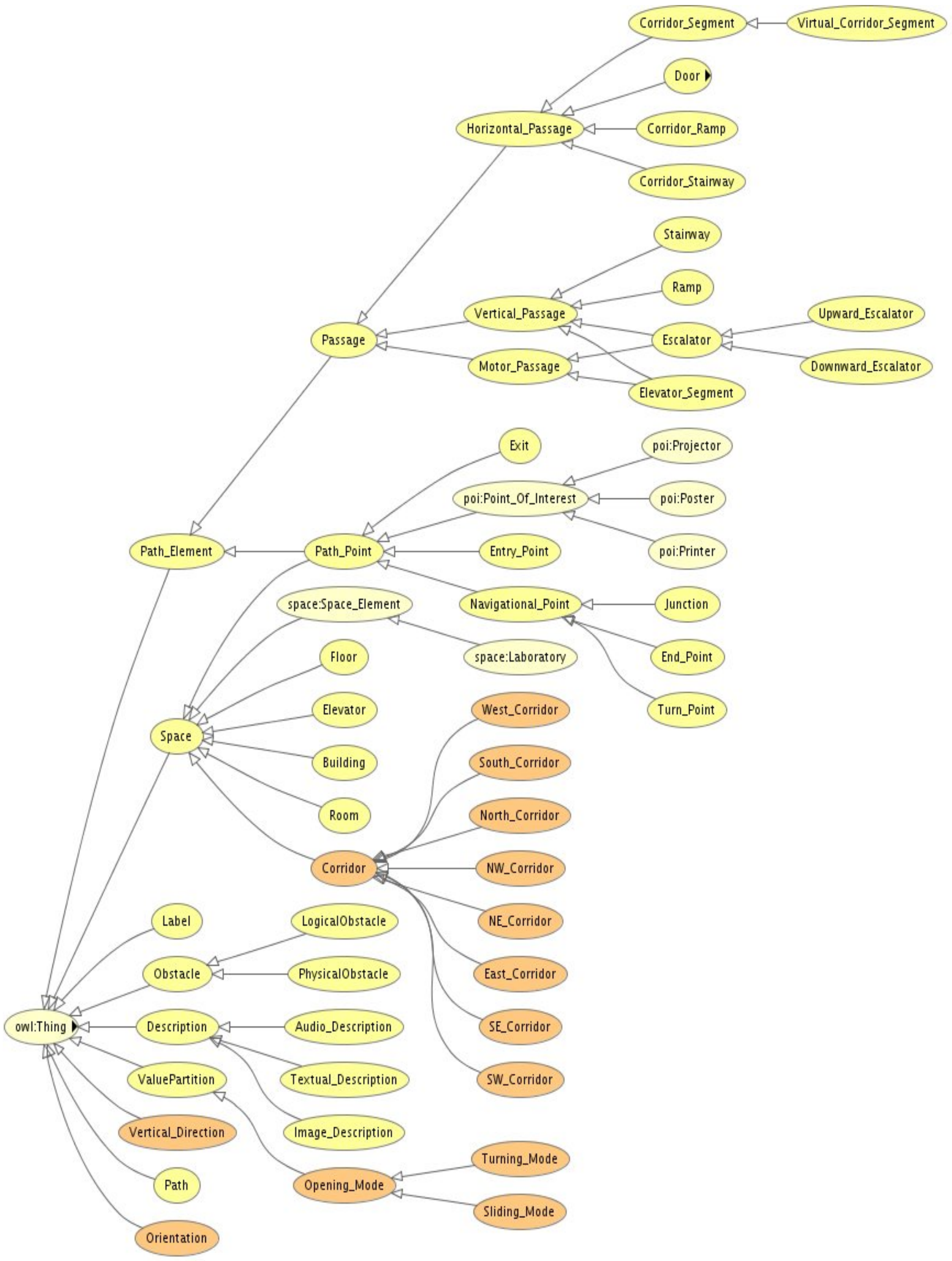


Figure 3.4: Diagram of Space Description Ontology

3.4 User Tracking & Location Modeling Ontology

3.4.1 Classes

This subsection will describe the classes of the tracking ontology and will state their utility and importance for the system functionality. In Figure 3.5 below, the diagrammatic representation of ontology's classes is depicted, expressing the hierarchy and inheritance that exist between general classes and their subclasses. The highlighted classes take specific values from predefined enumerations (see Figure 3.5).

- ➢ LocationDescription: LocationDescription is the general class which expresses the type of relation between users and path elements or POIs.
  - o DynamicDescription: DynamicDescription class, subclass of LocationDescription, contains all the dynamic types of relations, such as towards, away from etc.
  - o StaticDescription: StaticDescription class, subclass of LocationDescription, contains all the static types of relations, such as at, inside etc.
- ➢ Motion: Motion is the general class for modeling user's movement.
  - o Direction: Direction is the general class for modeling user's movement orientation.
    - ▪ ClockDirection: subclass of Direction, expresses user's movement according to the rotation of clock hands.
    - ▪ CompassDirection: subclass of Direction, expresses user's movement according to a magnetic compass.
    - ▪ RelativeXYDirection: subclass of Direction, expresses user's movement according to the X and Y axis.
    - ▪ Relative0Direction: subclass of Direction, expresses user's immobility.
    - ▪ RealtiveZDirection: subclass of Direction, expresses user's movement according to the Z axis.
  - o MotionState: MotionState class describes user's movement state.
    - ▪ Standing: subclass of MotionState, expresses user's immobility state.
    - ▪ Walking: subclass of MotionState, expresses user's moving state.
- ➢ Preferences: Preferences is the general class for modeling user's preferences.

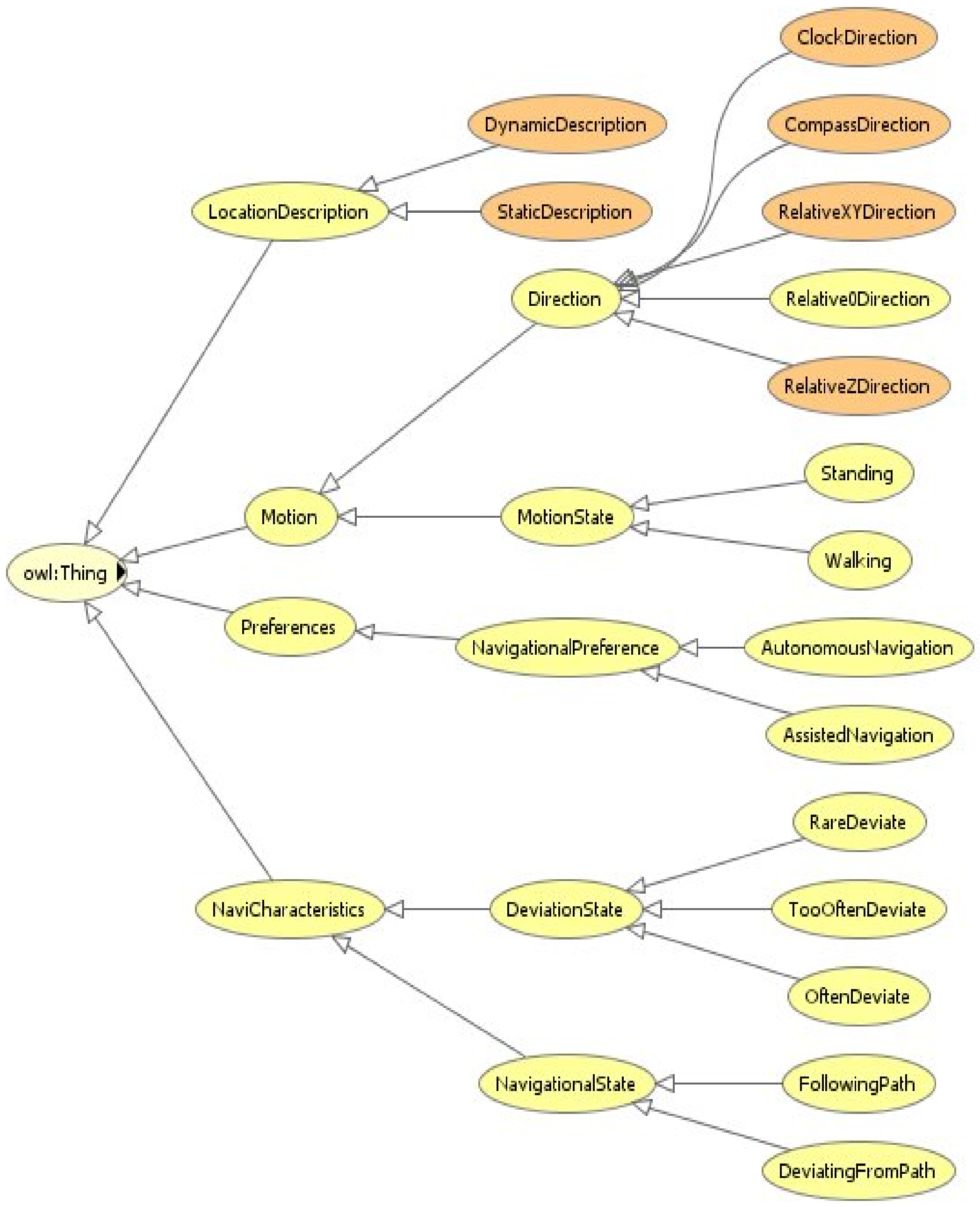


Figure 3.5: Diagram of User Tracking & Location Modeling Ontology

- NavigationalPreferences: NavigationalPreferences is the general class for modeling user’s navigation preferences.

        - AutonomousNavigation: subclass of NavigationalPreferences, expresses user’s navigation without system’s tracking procedure.
        - AssistedNavigation: subclass of NavigationalPreferences, expresses user’s navigation with the aid of system’s tracking procedure.
- NaviCharacteristics: NaviCharacteristics is the general class for modeling user’s navigation characteristics and information.
    - DeviationState: DeviationState is the general class for modeling user’s state according to the number of times he/she has deviated from the proposed path.
        - RareDeviate: subclass of DeviationState, contains users who belong to the category of 0 to 5 deviations from proposed path.
        - OftenDeviate: subclass of DeviationState, contains users who belong to the category of 6 to 10 deviations from proposed path.
        - TooOftenDeviate: subclass of DeviationState, contains users who belong to the category of 11 and above deviations from proposed path.
    - NavigationalState: NavigationalState is the general class for modeling user’s state according to his/her navigation behavior.
        - FollowingPath: subclass of NavigationalState, contains users who are following the proposed route.
        - DeviatingFromPath: subclass of NavigationalState, contains users who are deviating from the proposed route.

3.4.2 Properties

This subsection will describe the properties of the tracking ontology and will state their utility and importance for the system functionality.

- activates: The user of the system has the ability to activate either assisted navigation, in which system tracks user, or autonomous navigation, in which system does not track user.
- hasDeviationState: Every user has a deviation state, as he/she deviates from proposed path rarely or not at all (RareDeviate), he/she deviates often (OftenDeviate) or too often (TooOftenDeviate).
- hasDirection: Every user has a direction, which can be compass (CompassDirection), according to the clock hands rotation (ClockDirection), on the XY axis (RelativeXYDirection), on the Z axis (RelativeZDirection) and immobility (Relative0Direction).
- hasMotionState: Every user has a motion state, Walking or Standing.
- hasXTimesDeviated: This specific property stores in the ontology the integer number of user deviations.

- isCurrently: This specific property stores in the ontology the user's navigation state, which could be FollowingPath if he/she follows the proposed path or DeviatingFromPath if he/she deviates.
- setsTrackingTime: This property gives the ability to the user to specify the time between the first deviation and the next system's alert for dynamic re-planning.
- reportsLock: User informs system for locked door.
- reportsMalfunction: User informs system for malfunction in motor passage (e.g. elevator).
- reportsObstacle: User informs system for an obstacle blocking his/her navigation passage (e.g. blocked corridor).
- hasRelativeLocation: This property stores in the ontology the relationship between a user or POI with a space or path element and vice versa.
- hasDynamicDescription: This property stores in the ontology the type of user's dynamic relationship (from DynamicDescription class) with a space or path element.
- hasStaticDescription: This property stores in the ontology the type of user's static relationship (from StaticDescription class) with a space or path element.
- inDynamicDescription: This property stores in the ontology the type of space or path element's dynamic relationship (from DynamicDescription class) with a user or a POI.
- inStaticDescription: This property stores in the ontology the type of space or path element's static relationship (from StaticDescription class) with a user or a POI.

In Figure 3.2, all the properties mentioned above have been gathered. Every property is characterized by a domain and a range. The property's domain is the relation's subject whereas the range is its object. The table below presents the domains and ranges of all ontology's properties.

Table 3.2: Object and datatype properties, their domains and ranges

Name	Domain	Range
activates	user:User	NavigationalPreference
hasDeviationState	user:User	DeviationState
hasDirection	user:User	Direction
hasMotionState	user:User	MotionState
hasXTimesDeviated	user:User	int
isCurrently	user:User	NavigationalState
setsTrackingTime	user:User	time

reportsLock	user:User	sdo:Door
reportsMalfunction	user:User	sdo:Motor_Passage
reportsObstacle	user:User	sdo:Exit sdo:Navigational_Point sdo:Passage
hasRelativeLocation	user:User poi:Point_Of_Interest sdo:Path_Element sdo:Space	user:User poi:Point_Of_Interest sdo:Path_Element sdo:Space
hasDynamicDescription	user:User poi:Point_Of_Interest	DynamicDescription
hasStaticDescription	user:User poi:Point_Of_Interest	StaticDescription
inDynamicDescription	poi:Point_Of_Interest sdo:Path_Element sdo:Space	DynamicDescription
inStaticDescription	poi:Point_Of_Interest sdo:Path_Element sdo:Space	StaticDescription

3.4.3 Individuals

This subsection will describe the individuals of the tracking ontology and will state their utility and importance for the system functionality. The individuals correspond to the values included in specified class enumerations, so instances of the classes are allowed to take values only from those enumerations.

For the Direction class of the ontology, the individuals consisting class enumerations, as depicted in Figure 3.6, are:

- Ahead, Back, Right, Left: are the possible directions that user can move on X and Y axis (enumeration of RelativeXYDirection class).
- Up, Down: are the possible directions that user can move on Z axis (enumeration of RelativeZDirection class).
- North, South, East, West, North_West, North_East, South_West, South_East: are the possible orientations that user can move according to a magnetic compass (enumeration of CompassDirection class).

- Clockwise, Counterclockwise: are the possible directions that user can move according (or reverse) to the way clock hands rotate (enumeration of RelativeXYDirection class).

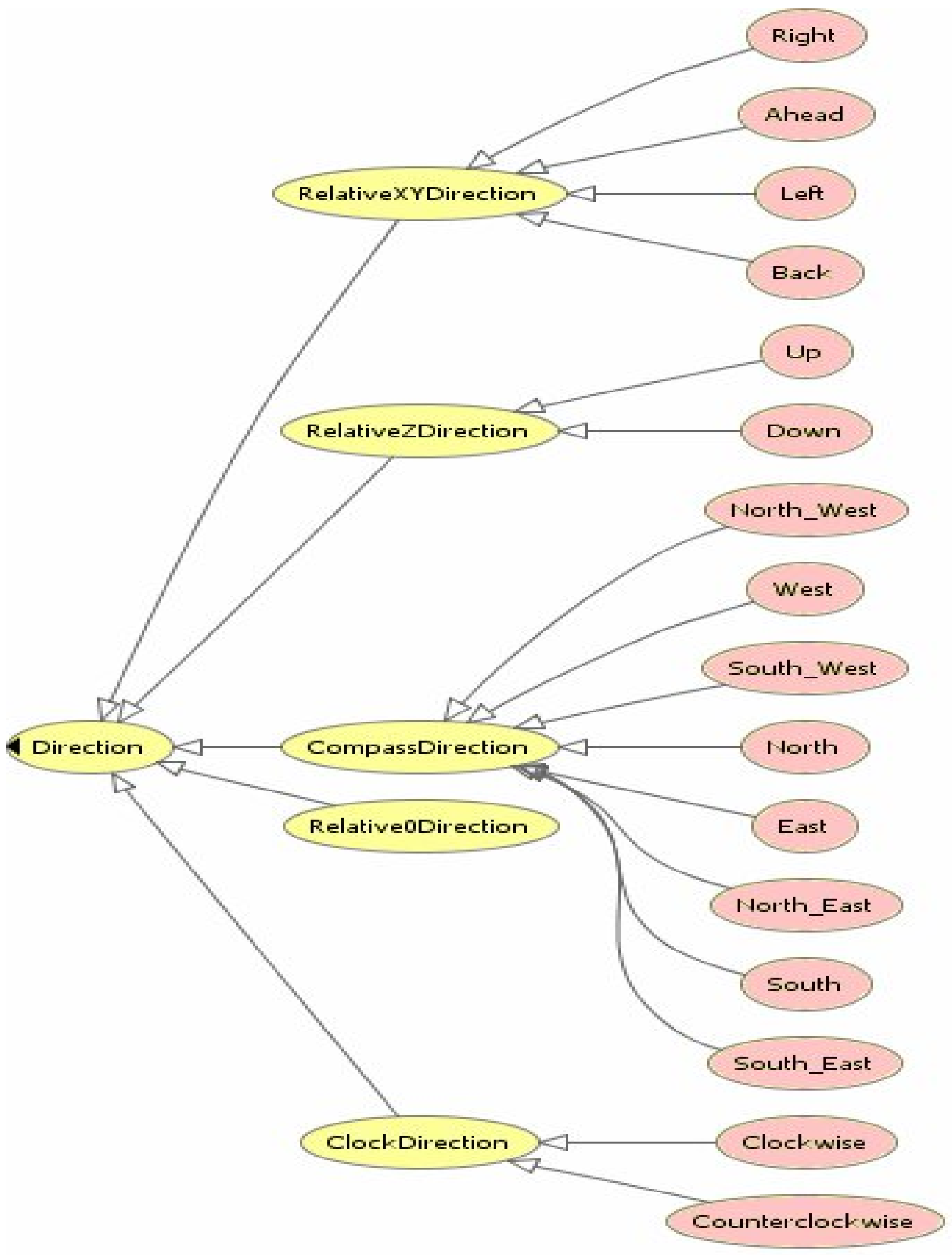


Figure 3.6: Direction Class Individuals

For the LocationDescription class of the ontology, the individuals consisting class enumerations, as depicted in Figure 3.7, are:

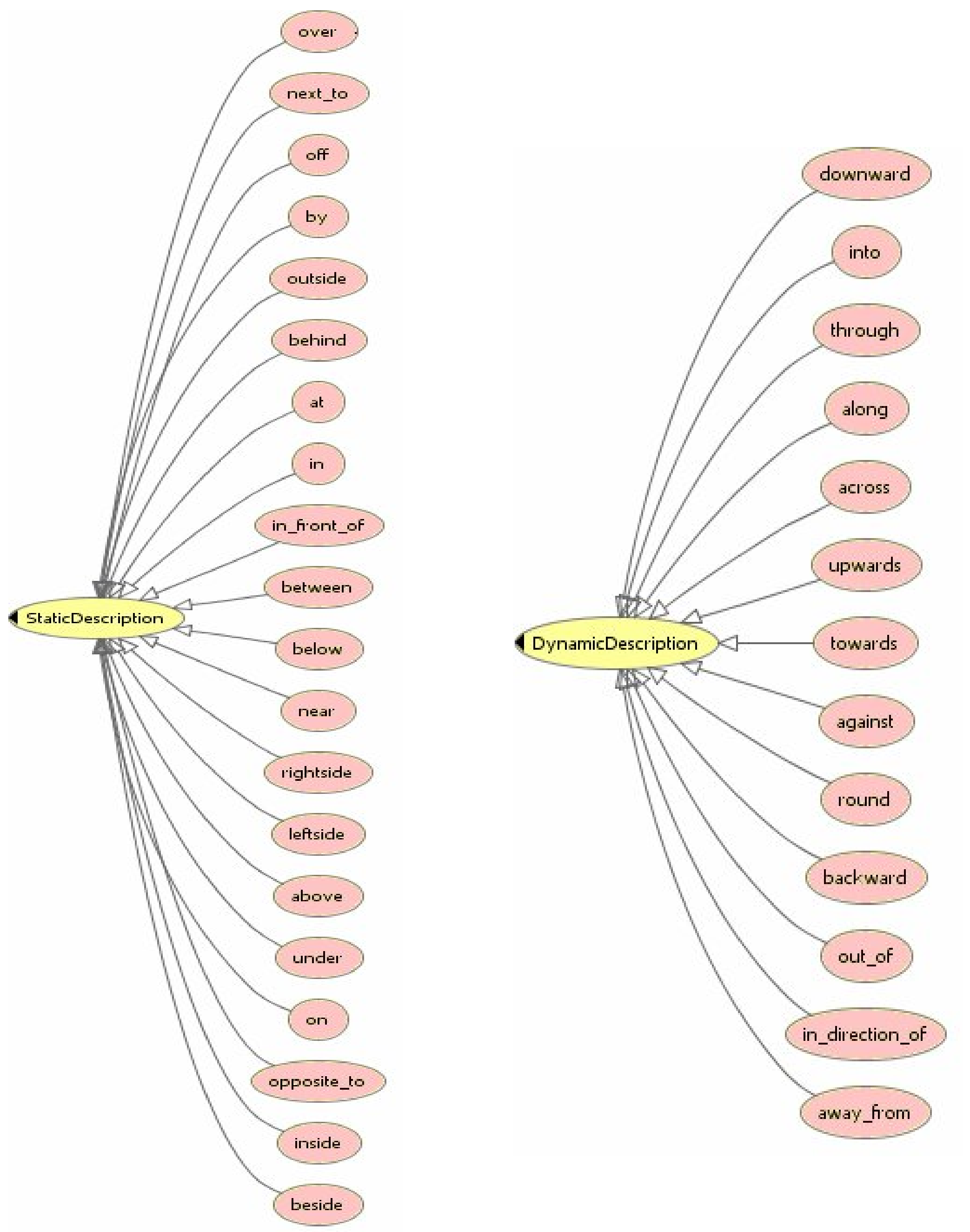


Figure 3.7: Static and Dynamic Description Classes Individuals

- next to, opposite to, inside, between, under, over, on, in, in front of, outside, above, by, below, near, at, beside, behind, leftside, rightside: are the possible descriptions for the static relation between user and path elements or POIs. An example of a static description of such a relation is: “User is walking *in* corridor segment”.
- Backward, across, toward, away from, out of, round, downward, into, along, against, upward, through, in direction of: are the possible descriptions for the dynamic relation between user and path elements or POIs. An example of a dynamic description of such a relation is: “User is walking *towards* an elevator”.

3.5 Path-Finding Algorithm

The k-shortest path problem is a variant of the shortest path problem, where one intends to determine k paths p1,…,pk (in order), between two fixed nodes. Each path pi should have cost greater or equal than pi−1, 1 < i-k, and the remainder paths between the fixed nodes should have cost at least equal to pk. More specifically, the k shortest paths problem is defined as follows: *“The determination of a set of paths $\{p_1, \ldots, p_k\}$ between a predefined pair of nodes s and t, which fulfil the shortest path criterion, as follows:*

$$len(p_{n-1}) \leq len(p_n) \text{ for every } n \leq k$$

where p_1 is the shortest path among the nodes s and t, as it would be calculated by a classic shortest path algorithm (e.g. Dijkstra).“

A basic element of the path-finding algorithm is not only the estimation of the shortest path, but the fulfilment of the criterions that are depended on user´s profile as well. The suggested algorithm for the implementation of the current application is based on a Yen’s project [1] and belongs to the general category of the divergent algorithms. The used code is a part of Mascopt’s project (Mascotte Optimization) and has developed an open source library for network optimization problems.

Mascopt library is an integrated framework and algorithm collection for networks and graphs. The library’s aim is the solution of network optimization problems. Mascopt library was used in the current system because it has already implemented an algorithm for finding the k-shortest paths of a graph. The specific algorithm is based on Yen’s algorithm and it can only find the k-shortest paths which do not contain loops in a graph whose edges do not contain negative weights.

Yen algorithm

Considering starting node s and terminal node t and a directed graph G(N,A), the path between s and t is defined as: $<s=u_1,a_1,u_2,\dots,a_{r-1},u_r=t>$, where $a_m \in A$, $u_n \in N$, m=1,…,r-1, n=1,…,r, A the number of directed edges of the G graph and N the number of the nodes. Considering, also, c_{ij} the length of the edge (i,j) ∈A, then the metric c(p) is the length sum of all the edges for the path p:

$$c(p) = \sum_{p} c_{ij}$$

For the calculation of the k shortest paths between s and t nodes, a set of paths $P^k = \{p_1,\dots,p_k\}$ should be computed with the following criterions:

- $c(p_i) \leq c(p_{i+1})$, for every i ∈{1,..,k-1}
- $c(p_i) \leq c(p)$, for every $p \in P \backslash P^k$, P is the number of all the paths from s to t.
- the p_i path has been calculated just before p_{i+1} for every i ∈{1,..k-1}

For the Yen's algorithm application, a specific structure should be defined in order to store the appropriate information for the k-shortest paths for the G(N,A) graph. This structure is a tree where a node could appear both as a parent and as a child node.

The basic point in the computation of the k-shortest paths is the calculation of the shortest paths between every node of a graph G and the terminal node t, and creating, as a result, a reversed tree T^*_t. This specific tree could be calculated using labelling algorithms by reversing all the edge directions, considering t node as the initial node and then executing a classic shortest path calculation.

3.6 Rules

The rule-based approach to realizing reactivity on the Web is an example of an easy to use Semantic Web technology. Compared with general purpose programming languages and frameworks, rule-based programming brings in declarativity, fine-grain modularity, and higher abstraction. Moreover, modern rule-based frameworks add natural-language-like syntax and support for the life cycle of rules. All these features make it easier to write, understand, and maintain rule-based applications, including for non-technical users.

There has been a large and exciting amount of recent progress in several aspects of semantic web rules, including fundamental knowledge representation for rules, fundamental knowledge representation to integrate rules with ontologies, translations between heterogeneous commercial rule languages and systems/engines, development of algorithmic techniques and open-source tools for inferencing and interoperability, standard proposals (including RuleML and SWRL) and pilot applications in e-business and other domains.

Reactive rules have the structure “ON Event IF Condition DO Action” and specify to execute the Action automatically when the Event happens, provided the Condition holds. Proactive (production) rules are of the form WHEN Condition DO Action and specify to execute the Action if an update to the local data base makes the Condition true. This shows that the similarities in the structure of these two kinds of rules come with similarities, but also some differences, in the semantics of the two rule paradigms.

3.6.1 Proactive Rules (SWRL Rules)

Semantic Web Rule Language is a proposal for a Semantic Web; combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog).The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. Both the antecedent (body) and consequent (head) consist of zero or more atoms. An empty antecedent is treated as trivially true, so the consequent must also be satisfied by every interpretation; an empty consequent is treated as trivially false so the antecedent must also not be satisfied by any interpretation. Multiple atoms are treated as a conjunction.

The most important SWRL rules designed for the system control are listed below. These rules are referred to the user motion modeling, to the user relative location modeling and the categorization of user’s profile according to the number of deviations. Note that prefix ‘user’ stands for classes of user profile ontology, ‘uto’ for classes or properties of user tracking ontology and ‘sdo’ for those of space description ontology. ‘swrlb’ prefix is referred to the SWRL build-in to handle some specific functions. More details about the full functionality which SWRL rules provide are presented in section 5.5.

- Deviate-category1:
  user:User(?x) ∧ uto:hasXTimesDeviated(?x,?y) ∧
  swrlb:lessThanOrEqual(?y, 5) ∧ uto:RareDeviate(?z)

→ uto:hasDeviationState(?x, ?z)
Description: If user x (user:User(?x)) has deviated y times (uto:hasXTimesDeviated(?x,?y)) and y is up to 5 (swrlb:lessThanOrEqual(?y, 5)), then user x has deviation state z (uto:hasDeviationState(?x, ?z)) where z is RareDeviate (uto:RareDeviate(?z)).

- Deviate-category2:
  user:User(?x) ∧ uto:hasXTimesDeviated(?x, ?y) ∧ swrlb:greaterThan(?y,5) ∧ swrlb:lessThan(?y,10) ∧ uto:OftenDeviate(?z)
  → uto:hasDeviationState(?x, ?z)
  Description: If user x has deviated y times and y is between 5 and 10 (swrlb:greaterThan(?y,5) ∧ swrlb:lessThan(?y,10)), then user x has deviation state z, where z is OftenDeviate (uto: OftenDeviate (?z)).

- Deviate-category3:
  user:User(?x) ∧ uto:hasXTimesDeviated(?x, ?y) ∧ swrlb:greaterThanOrEqual(?y, 10) ∧ uto:TooOftenDeviate(?z)
  → uto:hasDeviationState(?x, ?z)
  Description: If user x has deviated y times and y is 10 and above (swrlb:greaterThanOrEqual(?y, 10)), then user x has deviation state z, where z is TooOftenDeviate (uto: TooOftenDeviate (?z)).

- Standing:
  user:User(?x) ∧ uto:Relative0Direction(?y) ∧ uto:hasDirection(?x, ?y) ∧ uto:Standing(?z)
  → uto:hasMotionState(?x, ?z)
  Description: If user x has moving direction y (uto:hasDirection(?x, ?y)) and y is the state of immobility in relation with the (0, 0) point (uto:Relative0Direction(?y)), then user has motion state z (uto:hasMotionState(?x, ?z)), where z is Standing (uto:Standing(?z)).

- Moving1:
  user:User(?x) ∧ uto:ClockDirection(?y) ∧ uto:Walking(?z) ∧ uto:hasDirection(?x, ?y)
  → uto:hasMotionState(?x, ?z)
  Description: If user x has moving direction y and y is the direction according to clock hands (uto:ClockDirection(?y)), then user has motion state z, where z is Walking (uto: Walking (?z)).

- Moving2:
  user:User(?x) $\wedge$ uto:RelativeXYDirection (?y) $\wedge$
  uto:Walking(?z) $\wedge$ uto:hasDirection(?x, ?y)
  $\rightarrow$ uto:hasMotionState(?x, ?z)
  Description: If user x has moving direction y and y is the direction according to X or Y axis (uto: RelativeXYDirection (?y)), then user has motion state z, where z is Walking.

- Moving3:
  user:User(?x) $\wedge$ uto:RelativeZDirection (?y) $\wedge$
  uto:Walking(?z) $\wedge$ uto:hasDirection(?x, ?y)
  $\rightarrow$ uto:hasMotionState(?x, ?z)
  Description: If user x has moving direction y and y is the direction according to Z axis (uto: RelativeZDirection (?y)), then user has motion state z, where z is Walking.

- Moving4:
  user:User(?x) $\wedge$ uto: CompassDirection (?y) $\wedge$
  uto:Walking(?z) $\wedge$ uto:hasDirection(?x, ?y)
  $\rightarrow$ uto:hasMotionState(?x, ?z)
  Description: If user x has moving direction y and y is the direction according to a magnetic compass (uto: CompassDirection (?y)), then user has motion state z, where z is Walking.

- UserModel-PathElement:
  user:User(?x) $\wedge$ uto:StaticDescription(?y) $\wedge$ sdo:Path_Element(?z) $\wedge$
  uto:hasStaticDescription(?x, ?y) $\wedge$ uto:hasRelativeLocation(?x, ?z)
  $\rightarrow$ uto:hasRelativeLocation(?z, ?x)
  Description: If user x's position has static relation description y (uto:hasStaticDescription(?x, ?y)) with a path element z (uto:hasRelativeLocation(?x, ?z)) , then path element z has position relation with user x, too (uto:hasRelativeLocation(?z, ?x)).

- UserModel-Space:
  user:User(?x) $\wedge$ uto:StaticDescription(?y) $\wedge$ sdo:Space(?z) $\wedge$
  uto:hasStaticDescription(?x, ?y) $\wedge$ uto:hasRelativeLocation(?x, ?z)
  $\rightarrow$ uto:hasRelativeLocation(?z, ?x)

Description: If user x's position has static relation description y with a space element z, then path element z has position relation with user x, too.

DynamicDescription relation rules have the same logic with the last two SWRL rules containing the StaticDescription relations. It is very useful at this point to describe in detail the appliance of the user location modeling rules (i.e. the rules with the title 'UserModel-Space' and 'UserModel-PathElement'). The original intention for user's location modeling is to create a relation of the following type:

"John is inside roomA"

This expression implies the existence of a 3-ary relation, which will connect user class (John) with the static descriptions class (inside) and with the space element class (roomA). For that purpose, user tracking ontology uses three properties to connect each implicated class with the other two. Specifically, user class communicates with the class StaticDescription through the property hasStaticDescription and with the class Space through the property hasRelativeLocation. Respectively, Space class is connected with user class by hasRelativeLocation property and with StaticDescription class by inStaticDescription property. So to form the expression "John is inside roomA", the following relations must be created:

- hasStaticDescription(John,inside)
- hasRelativeLocation(John,roomA)
- hasRelativeLocation(roomA,John)
- inStaticDescription(roomA,inside)

Finally, is should be clear that the rules are also referred to Point_of_Interest class (instead of User class), to DynamicDescription class (instead of StaticDescription) and to Path_Element class (instead of Space class).

3.6.2 Reactive Rules (Jess Definition Rules)

Reactive rules are used for programming rule-based, reactive systems, which have the ability to detect events and respond to them automatically in a timely manner. Such systems are needed on the Web for bridging the gap between the existing, passive Web, where data sources can only be accessed to obtain information, and the dynamic Web, where data sources are enriched with reactive behaviour.

A category of reactive rules is the Jess definition rules. A Jess rule is something like an "if...then" statement in a procedural language, but it is not used in a procedural way. While "if...then" statements are executed at a specific time and in a specific order, according to how the programmer writes them, Jess rules are executed whenever their if parts (their left-hand-sides or LHS) are satisfied, given only that the rule engine is running. This makes Jess rules less deterministic than a typical procedural program. In other words, Jess rules are of the type: “on event, if condition is satisfied, then action is triggered”. The basic Jess rules designed in user tracking system are described below. They are targeting to cover two main theme categories, user feedback and user deviation. More details about the full functionality which Jess rules provide are presented in section 5.6.

-
  ```
  ( defrule lockRule
      (JessUser { reportsLock == TRUE } (specificUser ?usr) )
              =>
      (printout t  ?usr "reported lock" crlf)(calling DynamicReplanning ?usr)
  )
  ```
  Description: From the entity JessUser, if the boolean field ‘reportsLock’ becomes true, then print out which user (stored in variable usr) reported the locked door and call method DynamicReplanning(), with argument the specific user (i.e. the variable usr).

-
  ```
  ( defrule malfunctionRule
      (JessUser { reportsMalfunction == TRUE }  (specificUser ?usr) )
              =>
      (printout t  ?usr "reported malfunction" crlf)(calling DynamicReplanning
      ?usr)
   )
  ```
  Description: From the entity JessUser, if the boolean field ‘reportsMalfunction’ becomes true, then print out which user reported the malfunction of the motor passage and call the method ‘DynamicReplanning()’, with argument the specific user.

-
  ```
  (defrule obstacleRule
      (JessUser { reportsObstacle == TRUE }  (specificUser ?usr) )
              =>
      (printout t  ?usr "reported obstacle" crlf)(calling DynamicReplanning
      ?usr)
  )
  ```
  Description: From the entity JessUser, if the boolean field ‘reportsObstacle’ becomes true, then print out which user reported the path obstacle and call the method ‘DynamicReplanning()’, with argument the specific user.

-
  ```
  (defrule deviationRule
      (JessUser { deviates == TRUE }
      (specificUser ?usr) )
              =>
      (printout t "Deviation Detected for "?usr crlf)(calling
      DynamicReplanning ?usr)
  )
  ```
  Description: From the entity JessUser, if the boolean field ‘deviates’ becomes ‘true’, then print out which user deviates and call the method ‘DynamicReplanning()’, with argument the specific user.

Chapter 4

System Implementation & Conventions

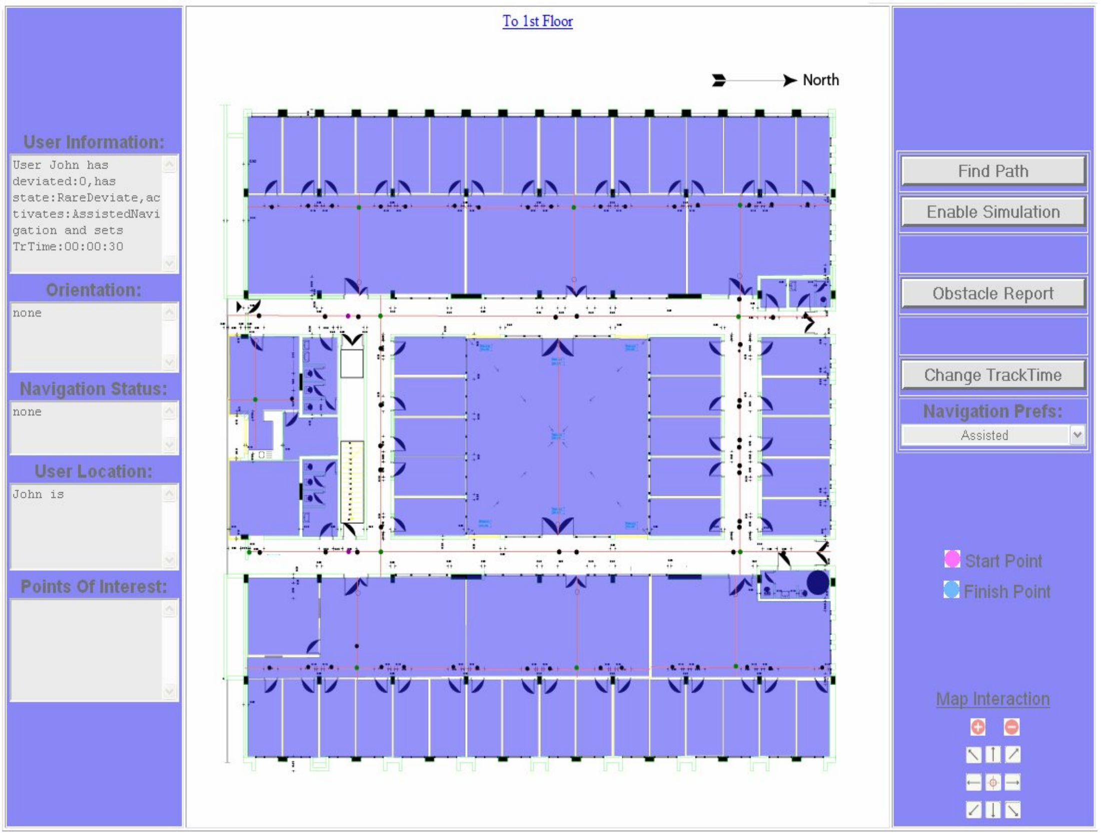
This chapter is an effort to present the full functionality of the integrated demonstration application, to describe the front and back-end conventions and to thoroughly analyze the system structure and the information flow during a selected scenario. It should be clear that the specific demonstration system is a direct application of the tracking system component and is not referred to the complete Contextual Guide.

4.1 Full System Functionality

As a part of the Contextual Guide, the tracking system application will have the intention to represent on user's map his/her current position. An arrow will inform about the movement direction and, in addition, user will have access to various data referring to the location relation with space or path elements and his/her profile updates.

An exception made in this particular application is the movement simulation, a process in which user can represent his/her position by clicking on the map. This change was made purely for testing reasons and in the absence of the location sensing system. As a result the tracking system translates every user click on the map to the corresponding coordinates. After this process is completed, the tracking system has the appropriate behavior, as if it was a part of the Contextual Guide.

In Figure 4.1, the completed GUI for the tracking system application is depicted. The user's main screen is separated in three thematic frames; the left frame is the information menu, which presents data about five information categories: user profile, user orientation, user navigation status, user location and POIs descriptions. This particular frame is updated dynamically, every time user proceeds to an action that influences one of the categories above.



User Information:
```
User John has
deviated:0,has
state:RareDeviate,ac
tivates:AssistedNavi
gation and sets
TrTime:00:00:30
```
Orientation:
```
none
```
Navigation Status:
```
none
```
User Location:
```
John is
```
Points Of Interest:

Navigation Prefs: Assisted

Figure 4.1: Graphical User Interface of the Tracking System demonstration application

The centered frame includes the map of the region on which the contextual guide is applied. In the specific demonstration application the map is the floor plan of the ground and first floor of ICS-FORTH. Initially, user observes the map intact until he/she selects a desired destination and then the system highlights in green color the proposed route. The user also has the ability to switch between the two building floors (ground and 1st floor) by clicking on the link at the top of the map. On the top right of the frame an arrow indicates the North direction, in order for the user to understand the orientation information box on the left frame.

Finally, the right frame includes all the interaction choices of the tracking system; which are: the 'Find Path' button, the 'Enable Simulation' button, the ‘Disable Simulation’ button in case simulation is already enabled, the ‘Obstacle Report’ and the ‘Reset Reports’ button, the ‘Change TrackTime’ button, the NavigationPrefs drop down menu and the buttons for the map interaction (e.g. zooming and panning).

4.1.1 User Interaction Functionality

- *Path-Finding*: User clicks on the ‘Find Path’ button and then selects start and destination points of the desired route, by clicking the corresponding navigation points on the map. The system takes as parameters start and destination points and forwards them to the path-finding library. The library returns all the path elements of the optimal route and the system highlights them on the map. In this way, the user has the ability to recognize the proposed path on the system’s map and to begin his/her walk.
- *Movement Simulation*: The original motivation of the Contextual Guide, as explained above, is to receive from the location sensing component the user’s coordinates. In the absence of location sensing and for stronger testing purposes, the movement simulation has been established. After selecting the desired path, user clicks on ‘Enable Simulation’ button and begins the simulation process of his/her walk on the proposed path. In more detail, user clicks on a location on the map, representing his/her current position, as every click simulates the user coordinates at the click’s exact position. System translates the click’s position coordinates to the corresponding path element’s id (using SVG technology) and forwards it to the application’s API. System’s API checks if path element’s id belongs to the proposed path; if it belongs, system permits user to continue movement simulation, by clicking on the next location. In different case, system triggers dynamic re-planning process to suggest new optimal path from the deviating position. If movement simulation is enabled, user can disable it by clicking on the ‘Disable Simulation’ button.
- *User Feedback*: After selecting the desired path, and during his/her movement simulation, user is able to click on ‘Obstacle Report’ button and report collision for the proposed path. Specifically, user has the option to report three types of collision, regarding the nature of the inflicted problem, by clicking on the collision point on the map. The three types are a locked door, a malfunctioning motor passage and an obstacle at a path element. System translates the click’s position coordinates to the corresponding path element’s id and forwards it to the application’s API. The API recognizes the type of collision according to the reported id and examines whether the collision really blocks user from following the path or not. If

system validates that the collision interrupts user's path, it triggers dynamic re-planning process to suggest new optimal path from user's current position. In other case, it just stores the collision point and takes it into consideration at the next path suggestion. If user has reported one or more collision points, he/she has the ability to remove them by clicking on the 'Reset Reports' button. Besides that, in the case of the integrated Contextual Guide, a system administrator will be responsible to check whether the reported obstacles are valid and if they are not, remove their system registration.

- *TrackTime Adjustment:* As TrackTime is defined the time that intercepts between a reported deviation and the next system's alert for new deviation. Suppose, a user who has deviated form proposed path, the system has dynamically re-planned his/her route, but user continues deviating for his/her own reasons. In this case, TrackTime is referred to the time that the system should wait between the initial deviation and the next dynamic re-plan process. User is able to adjust the TrackTime at any point by clicking the 'Change TrackTime' button and avoid the continuous alerts and re-plans in case of on purpose deviation.
- *Navigation Preferences:* User is able to click on the drop-down menu and select between Autonomous and Assisted navigation. During Autonomous navigation, user executes the movement simulation, the system tracks user movement, but has disabled the process of dynamic re-planning. In other words, system models user location in space but if he/she deviates from proposed path, system does not re-plan it. In Assisted navigation, the system includes the full tracking functionality, as system dynamically triggers re-plan process in case of deviation detection.
- *Map Interaction:* User has the option to interact with the application's map, by zooming in and out and moving it to eight directions: up, down, left, right, up-left, up-right, down-left and down-right.

4.1.2 User Information Functionality

- *User Profile:* User profile information box presents profile data which are referred to five categories; user registered name, number of user deviations, deviation state, navigation preferences and the adjusted TrackTime. Deviation state can be on of RareDeviate, OftenDeviate and TooOftenDeviate and navigation preference can be one of Assisted and Autonomous. When the system application initializes for a newly registered user, it categorizes him/her at the RareDeviate state and it enables the Assisted navigation preference by default. An example of user profile information box is: “User John has deviated 0 times, has state: RareDeviate, activates: AssistedNavigation and sets TrackTime at 00:01:00”.

- *Orientation:* Orientation information box states the system calculated user direction on the map and the MotionState in which he/she belongs. User can have one of the following directions; according to a magnetic compass (CompassDirection), according to the clock hands rotation (ClockDirection), on the XY axis (RelativeXYDirection), on the Z axis (RelativeZDirection) and immobility (Relative0Direction). The possible motion states are the Standing, if user is not moving, and Walking, otherwise. An example of orientation information box is: “moving North and has motion state Walking”.
- *Navigation Status:* Navigation status information box states whether user follows the proposed path, or he/she is currently deviating. The two different messages are: “is currently FollowingPath” and “is currently DeviatingFromPath”.
- *User Location:* User location information box presents the 3-ary relation between users, dynamic or static descriptions and space or path elements. An example of user location information box is: “John is inside Room _0_R44 on Floor _0 in Virtual_Corridor_Segment VCS-_0_R44E1P-_0_R44E2P”. This message explains that user John has static relation description ‘inside’ with space element Room _0_R44, static relation description ‘on’ with space element Floor _0 and static relation description ‘in’ with Virtual_Corridor_Segment VCS-_0_R44E1P-_0_R44E2P.
- *Points of Interest:* When user enters space or path elements which include points of interest, such as exhibits or posters, system highlights them on the map and user is able to select them and be informed about the description of the chosen POI. This data is browsed on the Points_of_Interest information box. Apart from the textual description given, the box prints the static relation description between user and POI. An example of Points_of_Interest information box is: “Poster5: CGIE-Contextual Guide for Indoor Environments, on John right-side”.

4.1.3 Dynamic Re-planning Functionality

Dynamic re-planning procedure is strongly connected with path-finding library and the incorporated algorithm. The specific library takes as arguments path points (e.g. junctions, exits), so the most important issue that re-plan process has to deal with is the correct calculation of these arguments. The arguments are referred to the start and destination points of user’s route. Of course, the calculation of the destination point is trivial, because it is the same destination as before dynamic re-plan triggering.

However, the specification of the starting point is a little more complicated. The general confrontation of the issue is to retract the path element that user has static relation with, before the trigger of dynamic re-planning. For this purpose system takes this information from the RelativeLocation property of the ontology model. If the retracted path element is a path point, then it is directly assigned as argument to the path-finding library. In case the element is a passage, further calculations must be executed.

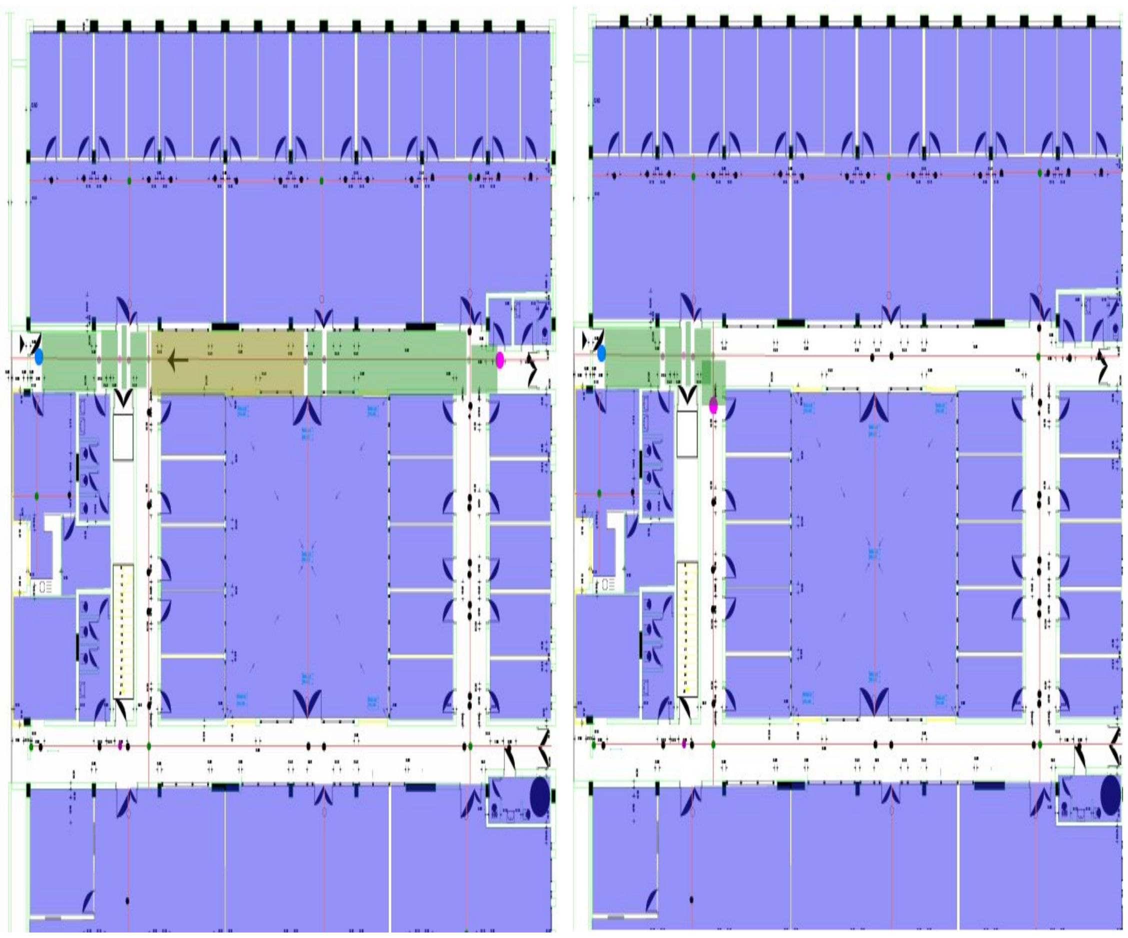


Figure 4.2: Tracking System representation before and after Dynamic Re-planning triggering

As mentioned before, passages connect two path points, so if a user has relative location relation with a passage before dynamic re-planning, the candidate starting points are the two path point of the specific passage. The most common idea is to adopt as starting point the end path point of the user's passage. For example, if John was in corridor segment CS-Point1-Point2 (the id of the specific corridor segment) before dynamic re-planning and during navigation he had overlapped Point1, the end point of CS-Point1-Point2 is Point2. If, on the other hand, John was moving reversely and had overlapped Point2, the end path point of the passage would be Point1. After calculating the end point of the passage according to user's movement, system puts it as second argument for the path-finding library.

There are two different occasions in which the system triggers dynamic re-plan; the first is user deviation. When system receives coordinates that correspond to a path element not in the proposed path or already overlapped by the user, it suggests that user has deviated from the proposed path. The second occasion is the user feedback; when user finds an obstacle blocking his/her navigation, reports it to the tracking system. The obstacle could be a locked door, a malfunctioning motor passage or a blocked path element. In both occasions, tracking system proposes a new optimal route.

Part A of Figure 4.2 shows the highlighted suggested path, user position and direction, while part B shows the new path after dynamic re-planning, as user turned left on the junction and deviated from the original route.

4.1.4 Context-Aware Guidelines and Filtering

Context-aware computing, as mentioned in subsection 1.2.5, is a new computing paradigm that has brought the possibility of exploring the dynamic context of the user in order to provide added-value services or to execute more and complex tasks. Building context-aware systems involves facing several design challenges to cope with highly dynamic environments and changing user requirements. Such challenges are mainly related to gathering, modelling, storing, distributing and monitoring context. These challenges justify the need for proper architectural support.

The introduced Guide is strongly characterised by a context-aware approach, especially the user profiling and path-finding system components. These components provide to the user a proposed route entirely based on his/her preferences (e.g. user prefers elevators instead of stairs), on privacy information (guide presents location and profile data of users only if their privacy setting permit it) and present event descriptions according to the research interests of users.

The user tracking component exploits the context in two different occasions; during the guideline information presentation and the illustration filtering of the points of interest. Specifically, while user navigates through a space element, the system locates the POIs inside the specific area. The Contextual Guide has the ability to filter the representation of these POIs on the map according to the user preferences. For example, if a user is interested in Information Science, the guide will hide the exhibits that are referred to other science (e.g. biology, physics, etc). With a similar procedure, the Contextual Guide is able to provide the navigation guidelines and user location and motion information in textual, symbolic format or both of them, according to user profile and preferences (e.g. if user is not able to read text; system represents his/her direction using only symbols).

4.2 Back-End Description

The back-end is the part of the application that performs all the essential tasks which are not apparent to the user and comprises the components that process the input and forwards the output for the front-end. In tracking application system, specifically, the back-end part is the Java API and the owl files which contain the ontology model.

4.2.1 API

The following subsection will present the packages included in the Application Program Interface of the tracking system and the main classes which consist this packages. A brief report about the functionality of the classes is also placed. The API was developed in Java, a high-level programming language developed by Sun Microsystems. Java is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web.

Package MainProgramCode

This package includes all the classes which have been developed in order to implement the main tracking system functionality.

- Profiling Class: This class is responsible for creating a new user instance and for updating important profile information. The main class functions are:
  - Creates a user instance, instantiating and initialising basic fields.
  - Sets navigational preferences for the user.
  - Sets system's tracking time (TrackTime) for the user.
- UserFeedback Class: This class takes a user's report of a path obstacle, a malfunctioning motor passage or a locked door and triggers the dynamic re-planning process only if the received feedback influences particular user's navigation. Specifically, the class checks if reported path element belongs to the proposed path, The main class functions are:
  - Receives user's report, checks if it blocks proposed path and then triggers dynamic re-planning.
  - Corresponds user with reported element in the ontology model.

  UserTracking Class: This class is responsible for user's location and motion modeling as well as the tracking process. It takes as arguments a specific user and the path element in which his/her current location belongs. Then, the class models user location and motion according to the given path element. The main class functions are:
  - Models user location and motion.
  - Checks for deviation from the proposed path.
  - Triggers re-planning if deviation is detected.
  - Alerts user for navigation status (normal navigation, deviation or successfully completed navigation).

  The first parameter that the class checks is whether the previous and the current coordinates received for the user's position are the same. If the coordinates are the same, the system categorises the user as having Standing motion state. In other case, system checks whether current user position is in the proposed path and, obviously, if he/she has not been in this position again. Supposing user is in the proposed path; this fact can lead in two conclusions; either user is following normally the suggested path or he/she has deviated in the past but dynamic re-planning was not triggered because of the TrackTime factor or the activation of the autonomous navigation preference by the user. Both options are examined by the system and the appropriate message is forwarded from the

UserTracking class (“Following Path” as message of first case and “Following Path after not Reported Deviation” of case option).

If the current user’s position does not belong to the proposed path or user has returned to a previous location (e.g. user is lost and is moving into circles), the class detects a path deviation. In this case, there are two possible system reactions which are influenced by the TrackTime factor. If the time of the last deviation added with the TrackTime is less than current time, system permits the dynamic re-planning triggering, or else, the system only forwards the appropriate message (“Deviating from Path” for the first case and “Dynamic Replanning not triggered (TrTime)” for the second case.). Finally, if current position corresponds to the destination of the suggested route, system informs user for successful completion of the navigation.

Before the message forwarding or the dynamic re-plan triggering, the UserTracking class executes the location and motion modeling. Firstly, program examines and categorises the path element to path point or passage. If the element is a passage, program assigns the passage’s orientation to the user’s direction property, if and only if, user is moving according to passage’s orientation. In other case, user’s direction is the reversed passage’s orientation. If the element is a path point, it has no orientation, so no user direction is assigned.

The location modeling is executed both for path elements and passages. Table 4.1 shows the static description relations that user can form with space or path elements. Apart from the relation that is created between user and current path element, some other relations are also calculated. The two basic factors which lead to advanced relations are the ‘entry_point’ and ‘exit’ path points. For the first factor, if user has passed from an ‘entry_point’, a static ‘inside’ relation between user and the room which the ‘entry_point’ leads, must be created. Likewise, if user has passed from an ‘exit’, program checks if user has relation with corresponding space element (e.g. room, for a room exit) and reforms the respective relations (e.g. reforms from ‘inside’ to ‘outside’ relation for a room exit). Finally, for current path element, program checks if it belongs to the same or to another floor and shapes the user-floor relation properly.

Table 4.1: Static Description relations between users and path or space elements

Path Element	**Static Description**	**Space Element**	**Static Description**
Corridor_Ramp	in	Building	inside

Corridor_Segment	in	Floor	on
Corridor_Stairway	in	Corridor	inside
Room_Corridor_Segment	in	Elevator	inside
Door	in_front_of	Room	inside/outside
Elevator_Segment	inside/outside		
Escalator	on		
Ramp	on		
Stairway	on/off		
Entry_Point	at		
Exit	at		
Navigation_Point	at		

- DynamicReplanning Class: This class re-plans the user's suggested navigation path from user's current position. The process for calculating the new starting point was analysed, in depth, in 'Dynamic Re-planning Functionality' subsection. The main class functions are:
  - Calculates user start and destination path points.
  - Calls path-finding algorithm with new path point arguments.
- GeneralAPI Class: This class executes all the aiding tasks which do not include access to the ontology model. Some of the main class functions are:
  - Saves and loads in owl files the ontology model.
  - Returns an instance given its id, from an input ontology class.
  - Returns the name of the parent class of input instance.
  - Removes all or any of the instances which are applied in user properties.
  - Returns 3-ary relations between Users, Spaces or Path Elements and static or dynamic description.
  - Calls path-finding library and returns all elements of the proposed path.
- navioAPI Class : This class is the general API class, the medium between API and frond-end, which provides access to all the tracking system packages and interconnects the classes of MainProgramCode package.

Package JessRules

This packege has been developed in order to implement the Jess language definition rules.

- JessHandler Class: Jess is a rule engine, a special kind of software that very efficiently applies rules to data. A rule-based program can have hundreds or even thousands of rules and Jess will continually apply them to data stored in its working memory. The rules might represent the knowledge of a human expert in some domain, a set of business rules, a technical specification, or the rules of a game. The working memory might represent the state of an evolving situation, the contents of a database, the status of a Web user's session, or a commercial account. In order to add knowledge inside the Jess working memory and run the Jess engine, the JessHandler class was developed. The main class functions are:
    - Creates a Jess rule engine and loads the rules.
    - Adds the java object of JessUser class in the working memory.
    - Runs the engine continuously and fires the rules that apply.
- JessUser Class: Java objects can be explicitly pattern-matched on the LHS of rules (as described in the previous chapter), but only to the extent that they are java beans. A java bean is really just a java object that has a number of methods that obey a simple naming convention for Java Bean properties. A class has a bean property if, for some string X and type T it has either or both of a method named getX which returns T and accepts no arguments; or, if T is boolean, named isX which accepts no arguments. A method named setX which returns void and accepts a single argument of type T. It should be noted that the capitalization is also important, for example, for a method named isChildOf, the property's name is childOf, with a lower-case C. Only the capitalization of the first letter of the name is important. For the above restriction of java object utility, the JessUser class was developed to create for every necessary java object its corresponding java bean. The main java bean objects are:
    - reportsObstacle, for the feedback of a path obstacle.
    - reportsMalfunction, for the feedback of a malfunctioning motor passage.
    - reportsLock, for the feedback of a locked door.
    - deviates, for the deviation detection.
- JessCaller Class: The Java interface jess.Userfunction represents a single function in the Jess language. To add new functions to the Jess language simply write a class that implements the jess.Userfunction interface, creating a single instance of this class and installing it into a jess.Rete object using Rete.addUserfunction(). The Userfunction classes can maintain state; therefore a Userfunction can cache results across invocations, maintain complex data structures, or keep references to external java objects for callbacks. There is no system type-checking on

arguments and values are self-describing. To implement the jess.Userfunction interface, only two methods must be implemented: getName() and call().The call() method does the business of the Userfunction. When call() is invoked, the first argument will be a ValueVector representation of the Jess code that evoked user function. JessCaller class was developed to implement jess Userfunction and call DynamicReplanning class after every rule triggering.

Package ProtegeCode

Protégé platform has the ability to create a java package which is related with the developed ontology. For each class of the ontology a java interface is created. This interface provides access to the OWL model and its elements, like classes, properties, and individuals. A factory class in the ProtegeCode establishes communication between this package and the MainProgramCode package.

4.2.2 Ontology Model

The class files created by the Protégé platform are connected with the corresponding ontology classes via URIs. In the tracking application, these URIs locate the owl files which include the ontologies models. The owl files that are used in the tracking system are:

- pois.owl: This owl file includes the ontology of the Points of Interest (POIs).
- space.owl: This owl file includes the ontology of the Space Elements.
- Profiles.owl: This owl file includes the ontology of the user profile.
- SDO.owl: This owl file includes the ontology of the location modeling.
- autocreated_instances.owl: This owl file includes the ontology of the instances created for the needs of the ICS-Forth modeling for the tracking application. More about the auto-created instances in the next subsection.
- UTO.owl: This owl file includes the ontology of the tracking component of the Contextual Guide.
- Instances.owl: This owl file includes the ontology of the necessary instantiated individuals before the initialization of the application.

The above ontologies embrace a specific hierarchy in order to function and interact properly. The SDO.owl ontology file imports both pois.owl and space.owl ontologies as they are important for the location modeling. Rules based on user profile (i.e. related to the path-finding process), need the Profiles.owl ontology and the profile includes location information about the user. As a result, SDO.owl imports Profiles.owl ontology and vice versa. It is obvious that the autocreated_instances.owl ontology should import SDO.owl, while user tracking ontology (UTO.owl) imports autocreated_instances.owl to form relations between users and space or path elements. Finally, the hierarchy ends, as Instances.owl imports UTO.owl ontology and has access to all necessary classes, properties and individuals.

4.3 Front-End Description

The front-end is the part of a software system that interacts directly with the user. In the tracking application system, specifically, the front-end part is the SVG map embedded in an html web page, which communicates with back-end using the JavaScript functionality.

4.3.1 SVG Functionality Capabilities

The embedded SVG files contain all the necessary data for the description of the ICS-FORTH ground plan, such as the rooms, the corridors, the elevators and stairs. Apart from the schematic representation of space, SVG file includes additional information, such as the coordinates of the path elements. SVG files are xml-based, so inside the description of the elements, any kind of important data can be added. The most important characteristic added in the description of every element is its unique id. All the ids were appointed with specific procedure, in order for the API to understand the type of path element that each id represents. An example of this procedure is the '_0_R15E' id, which is referred to a room exit, as the R stands for a room and the E stands for an exit. Moreover, the '_0' prefix assumes that this specific room exit belongs to the ground floor (floor 0) of the building.

Besides the coordinates and the id which is included for every element, SVG contains JavaScript event handlers for specific categories of path elements. All the path elements' ids are printed on the user interface while the system catches the 'onmouseover' and 'onmouseout' events, so user can be informed and click on the right point. The categories of SVG elements that handle the 'onclick' events for the specific application are:

- Corridor Segments: User clicks inside a corridor segment to either simulate his/her position or to report an obstacle blocking the path.
- Room Exit: User clicks on a room exit to simulate his/her position.
- Entry Point: User clicks on an entry point to either simulate his/her position or to report a locked door blocking the path.

- Elevator: User clicks inside an elevator to either simulate his/her position or to report a malfunction for this specific motor passage.
- Stairway: User clicks on a stairway to either simulate his/her position or to report an obstacle blocking the stairs.

Another important aspect about the SVG design is the definition of space for every path or space element. The most appropriate example to explain the issue is the corridor segments. Corridor segments are parts of a corridor which connect two path points. Because the corridors in the SVG files are represented as single lines, a more advanced approach was applied for the corridor segments, in order to handle situations in which the user clicks on the wider area of the corridors. As a result, corridor segments are polygons instead of simple lines and every click inside these polygons refers to a click inside the corresponding corridor segment. This approach is adopted for all the space and path elements that should be characterized by areas instead of simple lines or points on the map.

As mentioned before, SVG information is forwarded to the API with the aid of JavaScript which catches the user events. But how API and JavaScript establish communication? The answer is Java Applets. An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page. It is a software component that runs in the context of another program, for example a web browser and usually performs a very narrow function that has no independent use. When a Java technology-enabled browser is used to view a page that contains an applet, the applet's code is transferred to the system and executed by the browser's Java Virtual Machine (JVM).

The ‘java.applet’ package provides an API that gives applets some capabilities that applications do not have. Here are some other things that current browsers and other applet viewers let applets do:

- Applets can usually make network connections to the host they came from.
- Applets running within a Web browser can easily cause HTML documents to be displayed.
- Applets can invoke public methods of other applets on the same page.
- Applets that are loaded from the local file system have none of the restrictions that applets loaded over the network do.

A JavaScript API was developed in order to achieve the successful information flow between the SVG files and the application’s Java API. Three are the main functions which the JavaScript API executes:

- Start /End point specification: User’s clicks on the map are handled as input for the path-finding library (the clicked elements must be path points).

- User Feedback: User’s clicks on the map are handled as feedback for a path block (e.g. a locked door), so the clicked elements should be passages or entry points.
- Movement Simulation: User’s clicks on the map are handled as coordinates for the movement simulation process and every clicked element is considered as the user’s current location.

User enables every one of the categories above by clicking on the corresponding button of the GUI’s right frame.

4.3.2 Spatial Conventions

This subsection will describe the conventions made to fulfill two significant tasks, the orientation assignment of all horizontal passages on the SVG map and the calculation of the relation between the points of interest and the user.

Horizontal Passage Orientation

In the SVG file, horizontal passages have been created for the needs of the space ontology mapping. For example, in Figure, the red line depicts such a horizontal passage and more specifically, a corridor. In order for the system to recognize the user’s direction according to the magnetic compass when he/she enters a horizontal passage, all these passages should include orientation information in the ontology. When the mapping from the SVG file to the ontology is been executed, the system calculates the orientation of all the horizontal passages with a specific method which will be analyzed in the next paragraph.

As shown in Figure 4.3, all the corridors in the SVG file are characterized by a line which contains start (x3, y3) and end (x4, y4) coordinates. Moreover, the map includes an arrow with known start and end coordinates (x1, y1, x2, y2), which always points to the magnetic north (i.e. this arrow has north orientation). In order to determine the orientation of all the corridors, the angle between them and the arrow pointing the north must be calculated. Using the transformed type of the inner product shown below:

$$\arccos\theta = \frac{(x_2 - x_1)\cdot(x_4 - x_3) + (y_2 - y_1)\cdot(y_4 - y_3)}{\sqrt{\left[(x_2 - x_1)^2 + (y_2 - y_1)^2\right]\cdot\left[(x_4 - x_3)^2 + (y_4 - y_3)^2\right]}}$$

the angle between the two ‘vectors’ is computed and the declination of the corridor’s orientation form the north is also calculated.

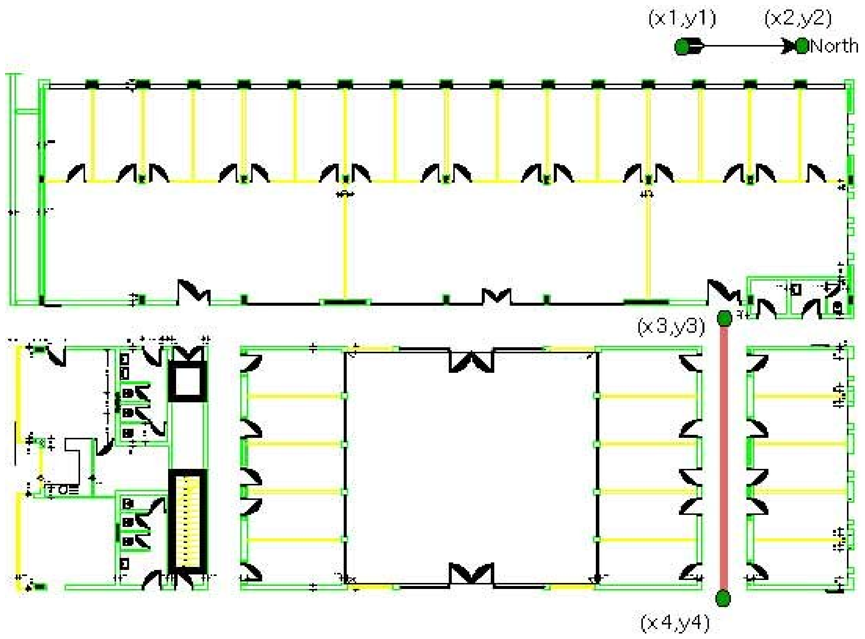


Figure 4.3: Coordinates of corridor vector and north indicator vector for the inner product calculation

Finally, to assign orientation to the corridors, a mapping between angles and compass directions is made, as shown in Table 4.2.

Table 4.2: Mappings between angles in degrees and orientations

Angle in degrees	**Orientation (CompassDirection)**	**Angle in degrees**
337.5 ≤	North	< 360
0 ≤	North	< 22.5
22.5 ≤	North-West	< 67.5
67.5 ≤	West	< 112.5
112.5 ≤	South-West	< 157.5
157.5 ≤	South	< 202.5
202.5 ≤	South-East	< 247.5
247.5 ≤	East	< 292.5
292.5 ≤	North-East	< 337.5

POI Location Calculation

The second task implicates the computation of the spatial relation type which exists between users and points of interest, when users enter a specific area. In order for the system to state these relation types, the user must enter horizontal passages because these path elements are designed as vectors inside the SVG map file and system can retrieve their start and end coordinates.

System calculates the projection coordinates (x', y') of the POI on the horizontal passage, as depicted in Figure 4.4 and then computes the angle between the passage and the line which is shaped between the POI and its projection. The computation of the angle is executed as before, using the transformation of the inner product type. Finally, the calculated angle can have only three values, 0 degrees, 90 degrees and 270 degrees, as the projection of the POI will always be vertical to the horizontal passage except if the POI is on the passage's vector. So, the three static description relations that can be shaped between POIs and users are: 'right-side' (for 90 degrees angle), 'left-side' (for 270 degrees angle) and 'in front of' (for 0 degrees angle).

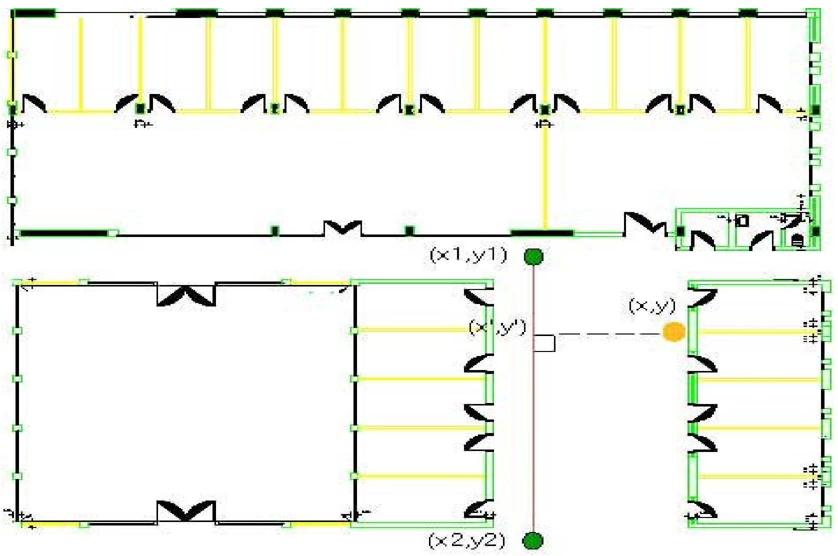


Figure 4.4: Calculation of the POI's projection on the corridor

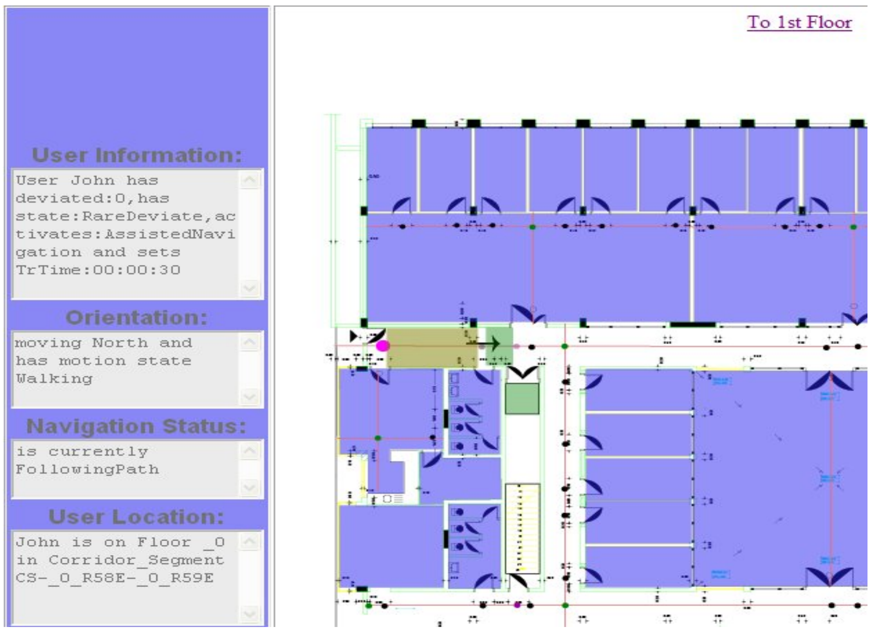
4.4 Information Flow

The proper communication between the components of the tracking system is a very important factor for the normal system functionality and this subsection will try to thoroughly describe the information flow between the distinct components. The description of the information flow will be presented by using a scenario in order to clarify, as much as possible, the communication procedure.

Scenario A

In the selected scenario, a specific user, John, enters the ground floor of the ICS-Forth from the location depicted by the purple dot on the map of Figure 4.5. John wants to reach the entrance of room R56 of the first floor, which is highlighted by the blue dot in Figure 4.5. His first action is to click on start and destination points, so the system takes from the SVG file the ids of these points and calls the path-finding library. The library returns the suggested path and the system highlights it on the map. John enables simulation of his movement and clicks his first position on the map (simulating his position coordinates).

Figure 4.5 shows the system's condition after the first click, which is updated in both the information and the map frame. In the map frame, system highlights the corridor segment in which John is and draws an arrow that informs for both exact position and direction. The box on the right of the arrow that belongs to the suggested path is the elevator. In the information frame, the Orientation, Navigation Status and User Location boxes have been updated. Orientation box informs John that he is moving North and his motion state is Walking. His Navigation Status is FollowingPath as John walks inside a highlighted area which belongs to the suggested route. Finally the User Location box informs John that he is on ground floor (Floor_0) and in corridor segment CS-_0_R50E-_0_R59E (i.e. the corridor segment which connects room's R50 exit and room's R59E).



User Information:

```
User John has
deviated:0,has
state:RareDeviate,ac
tivates:AssistedNavi
gation and sets
TrTime:00:00:30
```

Orientation:

```
moving North and
has motion state
Walking
```

Navigation Status:

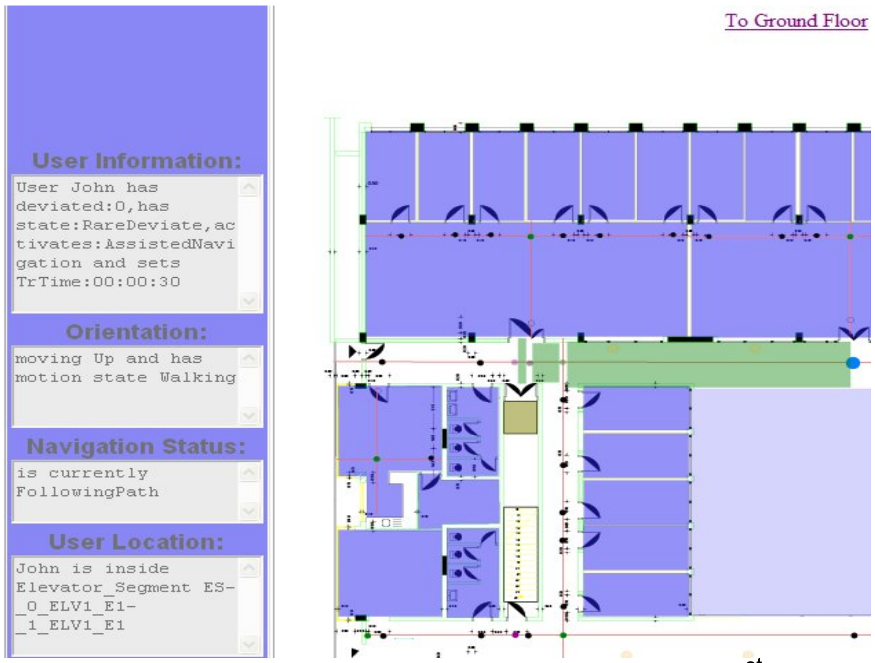
```
is currently
FollowingPath
```

User Location:

```
John is on Floor _0
in Corridor_Segment
CS-_0_R58E-_0_R59E
```

Figure 4.52: User starts navigation on the proposed path

In Figure 4.6, John has entered the elevator (the light green box) and is moving upwards to the first floor of ICS-Forth. By clicking on a vertical passage, the system switches the ground floor map view of the application with the first floor's map. The first floor's proposed path is highlighted until the destination point, which is depicted by the blue dot. The information frame is radically updated. The Orientation box shows new moving direction for John, the up direction, as the elevator transports him from a lower floor to an upper floor. User Information and Navigation Status boxes have not changed as user is still following the proposed path. Finally the User Location box has also changed, as John is informed about being inside elevator segment ES-_0_ELV1_E1-_1_ELV1_E1 (i.e. the elevator segment which connects ground floor elevator's ELV1 exit with first floor elevator's ELV1 exit). Moreover, floor relation between user and floor space element has been deleted, supposing that an elevator segment does not belong to a specific floor.


To Ground Floor

User Information:

```
User John has
deviated:0,has
state:RareDeviate,ac
tivates:AssistedNavi
gation and sets
TrTime:00:00:30
```

Orientation:

```
moving Up and has
motion state Walking
```

Navigation Status:

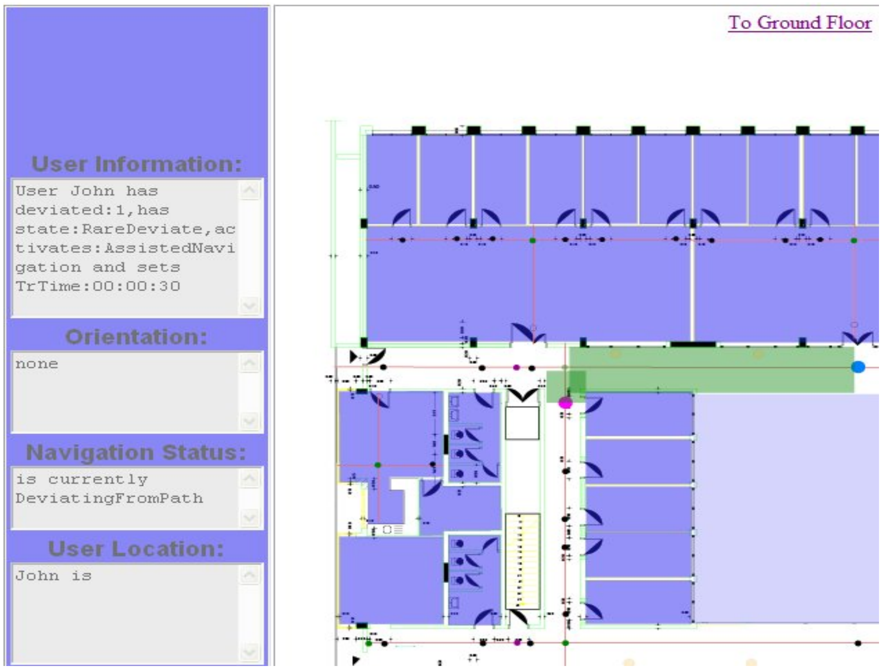
```
is currently
FollowingPath
```

User Location:

```
John is inside
Elevator_Segment ES-
_0_ELV1_E1-
_1_ELV1_E1
```

Figure 4.6: User is inside elevator and moves to 1st floor

Figure 4.7 shows the application condition after a user deviation. Specifically in the scenario, John after exiting the elevator and walking in the first floor's corridor, he turns right on the first junction he meets instead of going straight down the corridor. System detects that John has deviated from the proposed path and reactive rules trigger dynamic re-planning from John's deviation location to the same destination point. In the map frame, the purple dot depicts the new navigation starting point, while the blue depicts the pre-selected destination point. The information frame has updated both User Information and Navigation Status box. The tracking system has registered the deviation and changed the number of times John has deviated, adding one. Finally the Navigation Status box informs John that he currently deviates from the proposed path.



User John has deviated:1,has state:RareDeviate,activates:AssistedNavigation and sets TrTime:00:00:30

none

is currently DeviatingFromPath

John is

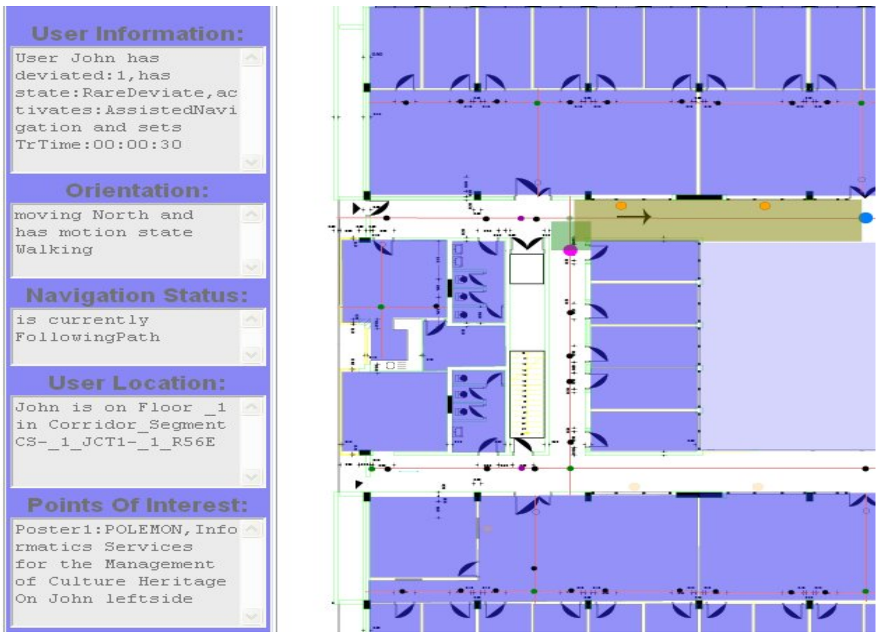
Figure 4.7: User had turned right on the first junction, system detected deviation and re-planned proposed path

In the last scenario screenshot, Figure 4.8, John continues his navigation following the highlighted path after the deviation detection. The black arrow indicates his location and direction towards the destination point (the blue dot). Inside the corridor segment, which is depicted with different color (light green), two orange dots have been drawn. System has recognized the existence of POIs inside the corridor segment that John has entered, so highlights the exact locations of these POIs, representing them as orange dots.

The POI recognition procedure is executed with a specific sequence of actions. As mentioned in the previous subsection, every point of interest is drawn at its exact location on the SVG map file, but with zero opacity. When John enters a passage or space element, the tracking system checks the space ontology to detect if any POIs belong in this specific element and returns the POIs ids. Then the system removes the transparency of these POIs from the SVG map file, so John is informed about all the interesting points inside his area.

As it is expected, back in the scenario, there are some significant updates on the information frame of the application. The User Information box has not changed and keeps indicating the previous deviation, while all the profile information remains intact. The Orientation box informs John that he is moving North according to the magnetic compass and has Walking motion state. The User Location box states that John now walks inside corridor segment CS-_1_JCT1-_1_R56E (i.e. the corridor segment which connects junction JCT1 and room's R56E exit) and navigates on the first floor of the building (Floor_1).

The last information box, Points of Interest, has been updated and presents data about the highlighted POIs. When user clicks on a desired POI, the system retrieves all the related textual description from the space ontology and presents it on this box. Moreover, John is informed about the location relation between him and the selected POI, which states that the POI is on John's left side. In Appendix A, another scenario is presented, in order to reveal more of the system's functionality.


User Information:

User John has deviated:1,has state:RareDeviate,activates:AssistedNavigation and sets TrTime:00:00:30

Orientation:

moving North and has motion state Walking

Navigation Status:

is currently FollowingPath

User Location:

John is on Floor _1 in Corridor_Segment CS-_1_JCT1-_1_R56E

Points Of Interest:

Poster1:POLEMON,Informatics Services for the Management of Culture Heritage On John leftside

Figure 4.8: User follows new path and is informed about a poster description

Chapter 5

Design & Implementation Tools

This chapter is an effort to mention, describe and highlight all the design and implementation tools which the thesis used to achieve the desired goal and finally integrate the demonstration application.

5.1 Ontology Web Language (OWL)

The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies, and is endorsed by the World Wide Web Consortium. OWL is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans [8]. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL. OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users [16].

- OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL.
- OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are

guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.

- OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold. Their inverses do not.

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

For the needs of the current thesis, the sublanguage selected was OWL DL. The main reasons of this choice (in contrast with OWL Lite), is the necessity for cardinality values more than 0 or 1 and the power that OWL DL gives in the area of reasoners for Description Logic (in contrast with OWL Full) [17].

5.2 Protégé 3.3.1

Protégé 3.3.1 is the platform used to design the User Tracking & Location Modeling ontology. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies [9]. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Furthermore, Protégé can be extended by way of a plug-in architecture and a

Java-based Application Programming Interface (API) for building knowledge-based tools and applications [10].

The Protégé platform supports two main ways of modelling ontologies - frame-based and OWL, each with its own user interface. The Protégé-OWL editor is an extension of Protégé that supports the Web Ontology Language (OWL), which is selected to develop our ontology. An important difference between Protégé and OWL is that OWL does not use the Unique Name Assumption (UNA). This means that two different names could actually refer to the same individual.

Protégé is a flexible, configurable platform for the development of arbitrary model-driven applications and components. Protégé has an open architecture that allows programmers to integrate plug-ins, which can appear as separate tabs, specific user interface components (widgets), or perform any other task on the current model. The Protégé-OWL editor provides many editing and browsing facilities for OWL models, and therefore can serve as an attractive starting point for rapid application development. Developers can initially wrap their components into a Protégé tab widget and later extract them to distribute them as part of a stand-alone application.

In addition to providing an API that facilitates programmatic exploration and editing of OWL ontologies, Protégé-OWL features a reasoning API, which can be used to access an external DIG compliant reasoner, thereby enabling inferences to be made about classes and individuals in an ontology.

A DIG (DL Implementation Group) compliant reasoner is a Description Logic reasoner that provides a standard access interface (a.k.a. the DIG interface), which enables the reasoner to be accessed over HTTP, using the DIG langauge. The DIG language is an XML based representation of ontological entities such as classes, properties, and individuals, and also axioms such as subclass axioms, disjoint axioms, and equivalent class axioms. The DIG langauge contains constructs that allow clients to "tell" a reasoner about an ontology and also "ask" a reasoner about what it has inferred, such as subclass releationships.

It is generally admitted that Protégé is not bug-free, but there is a lively mailing list which can help resolve issues and to which anyone can also submit bug reports or improvement suggestions.

5.3 Protégé-OWL API

The Protégé-OWL API is an open-source Java library for the Web Ontology Language (OWL) and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning based on Description Logic engines [27]. Furthermore, the API is optimized for the implementation of graphical user interfaces.The API is designed to be used in two contexts:

- For the development of components that are executed inside of the Protégé-OWL editor's user interface.
- For the development of stand-alone applications (e.g., Swing applications, Servlets, or Eclipse plug-ins).

The Protégé-OWL API is centered around a collection of Java interfaces from the model package. These interfaces provide access to the OWL model and its elements like classes, properties, and individuals. Application developers should not access the implementation of these interfaces (such as DefaultRDFIndividual) directly, but only operate on the interfaces. Using these interfaces programmer doesn't have to worry about the internal details of how Protégé stores ontologies. Everything is abstracted into interfaces and the code should not make any assumptions about the specific implementation. The most important model interface is OWLModel, which provides access to the top-level container of the resources in the ontology. OWLModel can be used to create, query, delete resources of various types and return objects for specific operations.

OWL and RDF resources are globally identified by their URIs, such as http://www.owl-ontologies.com/travel.owl#Destination. However, since URIs are long and often inconvenient to handle, the primary access and identification mechanism for ontological resources in the Protégé-OWL API is their name. A name is a short form consisting of local name and an optional prefix. Prefixes are typically defined in the ontology to abbreviate names of imported resources. The Protégé-OWL API makes a clear distinction between named classes and anonymous classes. Named classes are used to create individuals, while anonymous classes are used to specify logical characteristics (restrictions) of named classes.

5.4 RacerPro Reasoner

OWL-DL has its foundations in Description Logics, which are decidable fragments of First Order Logic. For a particular task, logic is decidable if it is possible to design an algorithm that will terminate in a finite number of steps. For example, in Description Logic it is possible to write an algorithm that calculates whether or not one concept is a subclass of another concept, which is guaranteed to terminate after a finite number of steps. Because an OWL-DL ontology can be translated into a Description Logic representation, it is possible to perform automated reasoning over the ontology using a Description Logic reasoner. A Description Logic reasoner performs various inferencing services, such as computing the inferred superclasses of a class, determining whether or not a class is consistent (a class is inconsistent if it cannot possibly have any instances), deciding whether or not one class is subsumed by another, etc. Some of the popular Descrioption Logic reasoners that are available: Racer, Fact, Fact++, Pellet. The reasoner used for the purpose of this thesis is Racer.

RACER stands for Renamed ABox and Concept Expression Reasoner. RacerPro is the commercial name of the software [28]. The origins of RacerPro are within the area of description logics. Since description logics provide the foundation of international approaches to standardize ontology languages in the context of the so-called semantic web, RacerPro can also be used as a system for managing semantic web ontologies based on OWL. However, RacerPro can also be seen as a semantic web information repository with optimized retrieval engine because it can handle large sets of data descriptions. Finally, the system can also be used for modal logics such as Km.

The semantic web is aimed at providing "machine-understandable" web resources or by augmenting existing resources with "machine-understandable" meta data. An important aspect of future systems exploiting these resources is the ability to process OWL documents (OWL KBs), which is the official semantic web ontology language. Ontologies may be taken off the shelf or may be extended for domain-specific purposes (domain-specific ontologies extend core ontologies). For doing this, a reasoning system is required as part of the ontology editing system. RacerPro can process OWL Lite as well as OWL DL documents (knowledge bases). Some restrictions apply, however. OWL DL documents are processed with approximations for nominals in class expressions and user-defined XML datatypes are not yet supported. The following services are provided for OWL ontologies and RDF data descriptions:

- Check the consistency of an OWL ontology and a set of data descriptions.
- Find implicit subclass relationships induced by the declaration in the ontology.
- Find synonyms for resources (either classes or instance names).
- Since extensional information from OWL documents (OWL instances and their interrelationships) needs to be queried for client applications, an OWL-QL query processing system is available as an open-source project for RacerPro.
- HTTP client for retrieving imported resources from the web. Multiple resources can be imported into one ontology.
- Incremental query answering for information retrieval tasks (retrieve the next n results of a query).

5.5 Semantic Web Rule Language (SWRL)

Semantic Web Rule Language (SWRL) is based on a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. SWRL is unique in being an extension of OWL DL, so that users of OWL DL can add rules to their ontologies and maintain clear semantics. SWRL includes a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sublanguages of OWL. A model-theoretic semantics is given to provide the formal meaning for OWL ontologies including rules written in this abstract syntax [22].

In common with many other rule languages, SWRL rules are written as antecedent-consequent pairs. In SWRL terminology, the antecedent is referred to as the rule body and the consequent is referred to as the head. The head and body consist of a conjunction of one or more atoms. Atoms in these rules can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values. The rules described in SWRL have the form below:

$$\textit{antecedent} \Rightarrow \textit{consequent}$$

where both antecedent and consequent are conjunctions of atoms written $a_1 \wedge \ldots \wedge a_n$. The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

Both the antecedent (body) and consequent (head) consist of zero or more atoms. An empty antecedent is treated as trivially true, so the consequent must also be satisfied by every interpretation; an empty consequent is treated as trivially false, so the antecedent must also not be satisfied by any interpretation. Multiple atoms are treated as a conjunction. Note that rules with conjunctive consequents could easily be transformed into multiple rules each with an atomic consequent. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., ?x). Using this syntax, a rule asserting that the composition of parent and brother properties implies the uncle property would be written:

parent(?x,?y) ⇒ brother(?y,?z) ⇒ uncle(?x,?z)

The proactive rules used for user categorisation, motion and location modeling were written in SWRL language.

5.6 Jess Rule Engine

Jess is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA. Using Jess, Java software can be built that has the capacity to "reason" using knowledge that is supplied in the form of declarative rules. Jess is small, light, and one of the fastest rule engines available. Its powerful scripting language gives access to all of Java's APIs.

Jess, or the Java Expert System Shell as it was first known, is rather more than the name suggests, comprising a general-purpose programming language which can access all Java classes and libraries. Mainly, Jess is a rule engine for the Java platform and in order for it to function; some logic must be specified in the form of rules using one of two formats: the Jess rule language or XML. When the rule engine runs, the rules are carried out. Rules can create new data, or they can do anything that the Java programming language can do. Although Jess can run as a standalone program, usually the Jess library is embedded in Java code and is manipulated it using its own Java API or the basic facilities offered by the javax.rules API.

Jess uses an enhanced version of the Rete algorithm to process rules. Rete is a very efficient mechanism for solving the difficult many-to-many matching problem [1]. A typical rule-based program has a fixed set of rules while the working memory changes continuously. However, it is an empirical fact that, in most rule-based programs, much of the working memory is also fairly fixed from one rule operation to the next. Although new facts arrive and old ones are removed at all times, the percentage of facts that change per unit time is generally fairly small. For this reason, the obvious implementation for the rule engine is very inefficient. In the Rete algorithm, the inefficiency described above is alleviated by remembering past test results across iterations of the rule loop. Only new facts are tested against any rule LHSs. Additionally, new facts are tested against only the rule (their left-hand-sides) LHSs to which they are most likely to be relevant.

Jess has many unique features including backwards chaining and working memory queries, and of course Jess can directly manipulate and reason about Java objects. Jess is also a powerful Java scripting environment, from which can create Java objects, call Java methods, and implement Java interfaces without compiling any Java code.

A Jess rule is something like an “if...then” statement in a procedural language, but it is not used in a procedural way. While “if...then” statements are executed at a specific time and in a specific order, according to how the programmer writes them, Jess rules are executed whenever their “if” parts (their left-hand-sides or LHSs) are satisfied, given only that the rule engine is running. This makes Jess rules less deterministic than a typical procedural program.

```
Jess> (defrule welcome-toddlers
          "Give a special greeting to young children"
          (person {age < 3})
                =>
          (printout t "Hello, little one!" crlf))
```

The above rule has two parts, separated by the "=>" symbol (which can be read as "then"). The first part consists of the LHS pattern (person {age < 3}). The second part consists of the RHS action, the call to println. The LHS of a rule consists of patterns which are used to match facts in the working memory, while the RHS contains function calls.

Jess can be used in many ways. Besides the different categories of problems Jess can be applied to being a library and used in many different kinds of Java programs. Jess can be used in command-line applications, GUI applications, servlets, and applets. The reactive rules used for catching user’s deviation and the obstacle report were written in Jess language and triggered by Jess rule engine.

5.7 SVG & Illustrator CS3

Adobe Illustrator CS3 offers many helpful features that bring new ease to creating and managing Scalable Vector Graphics (SVG) for mobile devices and the web. The demonstration application of the thesis was developed on Adobe Illustrator CS3 and the ground plans used were in SVG file format.

Scalable Vector Graphics (SVG) is a graphics file format and Web development language based on XML. SVG enables Web developers and designers to create dynamically generated, high-quality graphics from real-time data with precise structural and visual control. With this powerful new technology, SVG developers can create a new generation of Web applications based on data-driven, interactive, and personalized graphics. SVG aims to be to graphics what HTML is to page layout. SVG allows artists and developers to specify graphic images using an HTML-like markup language, allowing for graphics that not only take up much smaller amounts of bandwidth, but also allows scripting of the graphical elements for animation and interactivity.

SVG is a language for describing two-dimensional graphics and graphical applications in XML. It enables Web developers, designers, and users to move beyond the limitations of HTML and create robust visual content and interactivity through a simple declarative programming model. SVG is suitable for Web applications based on data-driven, interactive, personalized graphics from real-time data sources such as e-commerce systems and corporate databases [5]. Developers can customize SVG for many audiences, cultures, and demographics, no matter how the user interacts with the data.

SVG makes it possible to specify, using text, graphical images that appear on the page. For example, where a traditional graphic would need to specify every pixel of a rectangle, an SVG simply states that the rectangle exists, and specifies its size, position, and other properties. The advantages are many, including the ability to easily generate graphics (such as graphs and charts) from database information, and the ability to add animation and interactivity to graphics. Industry applications of SVG include mobile authoring, print based on XML page description including variable data printing, Web applications, and Geographic Information System (GIS) mapping.

SVG was introduced as an open standard by the World Wide Web Consortium (W3C) in 1999 for publishing animation and for interactive applications using vector graphics on the Web [29]. In 2004, a vast majority of the mobile phone industry chose SVG as the basis for its graphics platform. The features of SVG:

- *Small file size*: On average, SVG files are smaller than other Web graphic formats, such as JPEG and GIF, and are quick to download.
- *Display independence*: SVG images are always crisp onscreen and print at the resolution of your printer, whether it's 300 dots per inch (dpi), 600 dpi, or higher. You never get jagged edges due to pixel enlargement or anti-aliasing.
- *Superior color control*: SVG offers a palette of 16 million colors and supports ICC color profiles, sRGB, gradients, and masking.
- *Interactivity and intelligence*: Since SVG is XML-based, it offers unparalleled dynamic interactivity. SVG images can respond to user actions with highlighting, tool tips, special effects, audio, and animation.
- *Zooming*: Users can magnify an image up to 1,600% without sacrificing sharpness, detail, or clarity. Text stays text in SVG, images remains editable (within the source code) and, more importantly, SVG is searchable (unlike in raster and binary counterparts). There are no font or layout limitations, and users always see the image the same way you do.
- *Text-based files*: An SVG file is text-based, not binary. It is a "human readable" format much like HTML. Even a beginner can look at SVG source code and immediately make sense of the descriptive content relative to the graphic representation.

Chapter 6

Conclusion & Future Work

This final chapter will summarize the basic design and implementation characteristics of the thesis research, will describe a small efficiency evaluation and finally will present the ideas for future research and the motivation for an integrated Contextual Guide.

6.1 General concluding ideas

Current thesis was dedicated to present and describe a Contextual Guide for indoor environments and specifically, the user tracking component. It was also an attempt to support the lack of research in the field of user tracking, during navigation, as well as the modeling of both user location and motion.The first chapter introduced the subject of the master's thesis and presented briefly the main points of research and integration. Furthermore, a separated section was dedicated to inform the reader about specific terminology, in order to get familiar with useful definitions and background knowledge, essential for the comprehension of current thesis.

Chapter 2 presented and summarized the main characteristics of some important integrated navigation systems and mentioned not only their positive points but their weaknesses as well. Then, a small review of the literature related to location-based modeling systems was delivered and finally a small section about the innovative ideas, that this particular thesis introduces, were outlined. Chapter 3 presented the full system design and architecture, while chapter 4 described all the implementation techniques as well as the system information flow. Finally, in chapter 5 all the design and implementation tools, which have been used in the application, were mentioned.

In general, the Contextual Guide aims to provide a navigator for indoor environments, which will directly depend on the user profile, exploiting information about his/her physical and mental status and preferences. Using such kind of data, the guide will adapt the path-finding algorithm, through the aid of predefined rules, in order to provide the optimal and not necessarily the quickest route. This system approach gives to the application a pure human-centric base. Considering the fact that the system is designed to take full advantage of the semantic information, the final application is transformed to an exceptionally human friendly navigation system.

As far as the tracking component is concerned, the main architecture characteristics are summarized below. The research process separated the system in three distinctive layers, the ontology layer, the application program interface (API) along with the design rules and the graphical user interface (GUI). The functionality of the first layer, the user tracking, motion & location modeling ontology is summarised below:

- *User motion modeling*: The system recognises the type and orientation of user’s movement and categorizes it inside the ontology, in the appropriate class.
- *User position description*: The system expresses static and dynamic relations between user and path elements or POIs during his/her navigation, according to the proposed location descriptions of the ontology.
- *User profile information*: The system updates user profile according to three criterions, whether user follows or not the proposed path, whether he/she often or rarely deviates and whether he/she has activated assisted (user tracking process enabled) or autonomous (user tracking system disabled) navigation.

The second layer contains the application’s middleware API, which communicates with both the ontology and the GUI. In fact, it forwards the model’s information to the user interface and correspondingly updates the ontology according to user’s behaviour and actions. Very important is also its relation with the designed rules that facilitate control process.

The last layer represents the user interface and includes three different components:

- *User Interface*: Is referred to the html and JavaScript based features which are developed to present to every user the contextual guide application.
- *SVG based map modeling*: Is referred to the SVG map files which include the geographical representation of desired space and contain useful information, such as path element coordinates.
- *The GUI API*: Is referred to the JavaScript API which is developed to receive spatial information from the SVG, handle user actions and establish communication with the application’s middleware API.

Three are also the main categories in which the integrated system’s functionality can be distinguished:

- *User Interaction Functionality*: includes the Path-Finding part (user selects start and destination points of the desired route, system highlights the proposed path on the map), the Movement Simulation (user clicks on a location on the map, representing his/her current position, as every click simulates user coordinates) and User Feedback (user has the option to report a collision during the navigation on the proposed path, by clicking on the collision point on the map).
- *User Information Functionality*: includes the User Profile part (user profile data which are referred to user registered name, number of user deviations, deviation state etc), the Orientation part (user's direction on the map and his/her motion state), the Navigation Status part (whether user follows the proposed path or he/she is currently deviating), the User Location part (the 3-ary relation between users, dynamic or static descriptions and space or path elements) and the Points of Interest part (the description of a chosen POI).
- *Dynamic Re-planning Functionality:* is the procedure in which user's proposed path is dynamically re-planned from his/her current position and a new optimal path is printed on user's map. The two different circumstances in which the system triggers dynamic re-plan are: user deviation (when system receives coordinates that correspond to a path element not in the proposed route or already overlapped by the user) and user feedback (when user finds an obstacle blocking his/her navigation).

The innovative part of the thesis is the ability of the proposed system to track user's movement after suggesting the optimal path, to check if user follows it or to detect path deviation. If deviation is detected, the system alerts user and highlights a new valid path from the deviation location to the desired destination. Apart from user tracking and dynamic re-planning, the system has the ability to provide semantic and geographic information about the position of user in space and the direction of his/her motion. Furthermore, it can present all the points of interest in the nearby area of user position. Finally, the User Tracking Ontology's ability to create 3-ary relations between users, descriptions and objects, for example: "John is inside elevatorX ", is a very powerful contribution to simplify the location modeling and provide as much semantic and spatial information as possible.

6.2 Efficiency evaluation

This section will describe the evaluation of the system in specific scenarios and will present diagrammatically the main categories which generate time delay in the application. These main categories are described below:

- *Embedding SVG*: The application implementation selected SVG technology to support map representation. Every building floor demands a distinct ground plan stored in a separate SVG file. The load of all these files implicates a latency factor.
- *Loading OWL Model*: During the initialization of the system, the ontology model is loaded from the OWL files, implicating a latency factor.
- *k-Shortest Paths*: The path-finding process uses the k-shortest paths algorithm , which includes a specific complexity that corresponds to a latency factor.
- *SWRL Rules*: The execution of SWRL rules to categorize instances and to create 3-ary relation between users-descriptions-elements, during the movement simulation, imposes a latency factor.
- *Jess Rules*: The execution of JESS rules to trigger dynamic re-planning, during the movement simulation, imposes a latency factor.

The conditions of the system execution require the examination of the latency in two different cases, at the initialization process and during the execution of the system. These cases where confirmed by the testing scenarios , in which it was clear that both path-finding algorithm and SWRL rules were more time consuming during the initialization than during the execution of the application. SWRL rules are loaded in memory after their first evocation, so the next time they need less time. Moreover, the SVG file is stored in browser's cache and if navigation suggests the switch of the floor (i.e. the switch of the SVG file), the waiting time is substantially reduced. Finally the load of the ontology model from the owl files is only executed once, so this latency factor does not extend the execution time after the first initialization.

Figure 6.1 shows the diagrammatic representation of the latency factors during the application initialization. The greatest percentage, 35.97%, stands for the time required for embedding the SVG file and corresponds to approximately 3000 milliseconds (msecs). The next percentage, 28.48%, represents the time required to load the ontology model from the owl file and corresponds to 2375 msecs.

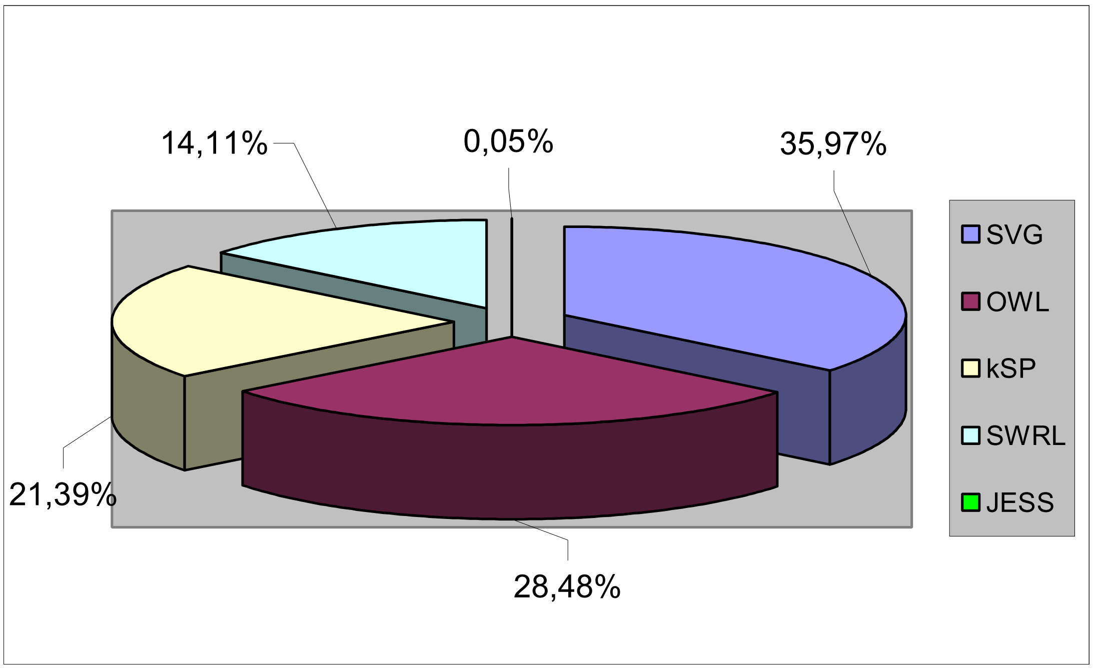


Figure 6.1: The percentage distribution of the latency during the system initialization

The third percentage, 21.39%, depicts the time needed by the k-shortest paths algorithm to return the optimal path for the user navigation and corresponds to 1784 msecs. The SWRL evocation represents the 14.11% of the overall delay, approximately 1177 msecs. Finally, the last percentage, 0.05%, stands for the triggering time of the JESS reactive rules and adds only 3.2 msecs delay to the application.

Figure 6.2 shows the diagrammatic representation of the latency factors during the execution of the application. The greatest percentage, 52.47%, stands for the time needed by the k-shortest paths algorithm to return the optimal path for the user navigation after a dynamic re-planning and corresponds to 1175 msecs.

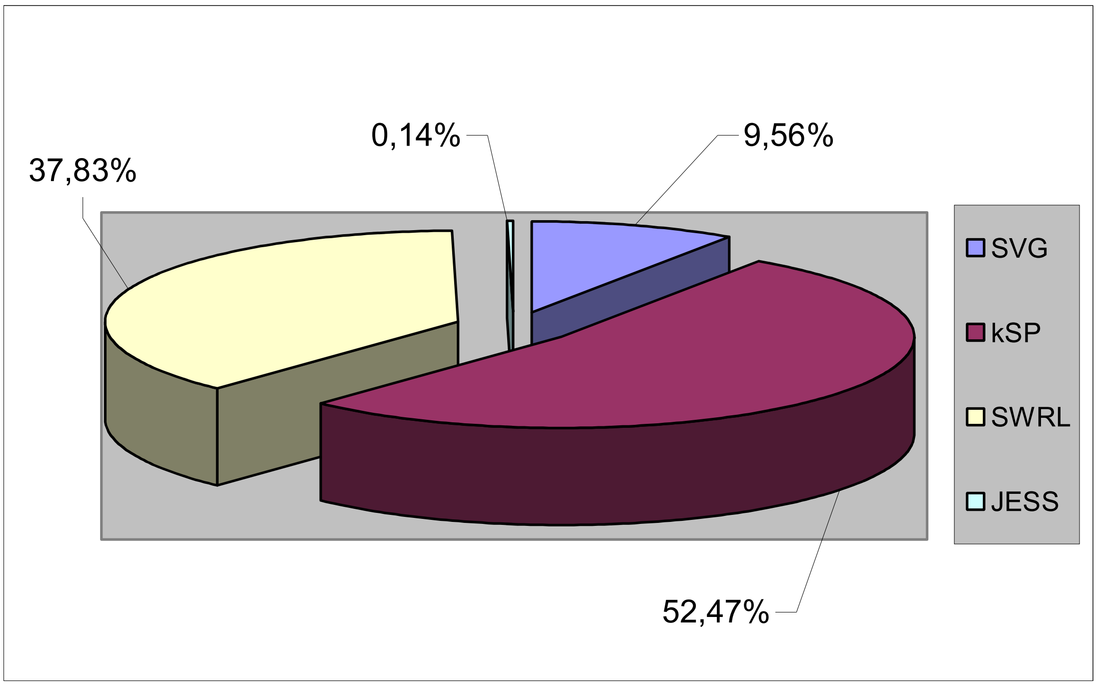


Figure 6.2: The percentage distribution of the latency during the system execution

The next percentage, 37.83%, represents the time required for the SWRL evocation, approximately 847 msecs. The third percentage, 9.56%, depicts the time required for switching floors and changing the SVG map files, corresponding to approximately 214 msecs. Finally, the last percentage, 0.14%, stands for the triggering time of the Jess reactive rules and adds only 3.2 msecs delay to the application.

From all the above results of the application's latency, some useful conclusions could be outlined. The significant point which could be stated is the average user waiting time from the system initialization until the first path suggestion, corresponding to 7159 msecs, about 7 seconds. This time includes the SVG (3000msecs) and the ontology model (2375msecs) loading process and the path-finding algorithm latency (1784msecs). The 7seconds seem to be inefficient waiting time; however this is an unstable opinion considering that the specific time is needed only for the system initialization.

In fact, the time consumed by the system's components during most of the execution process (i.e. the movement simulation) is only 847 msecs, less than a second. In detail, the system's most frequent executed functionality is the user's movement simulation. This process involves only the SWRL rules evocation latency, which is encountered at every user's click on the map (simulating his/her position). This latency corresponds to the SWRL percentage of Figure 6.2, approximately 0.85 seconds; time short enough not to be inconvenient for the user interaction.

Finally, the latency introduced by the system after every dynamic re-planning process corresponds to only 1178.2 msecs, which includes the path finding algorithm's estimated delay, about 1175 msecs, and the Jess triggering time, only 3.2 msecs. This latency is comparatively small and user barely realizes this delay during the execution flow.

6.3 Integration of a complete application

The original motivation, as described in previous chapters, is to create a semantics-based contextual guide for indoor environments. The contextual guide will be designed for pedestrian navigation regarding user's physical and mental profile. The design plan to fulfil the original expectations led the research team to divide the total study into three different research areas. The first area is occupied with the design and implementation of dynamic semantic profiles and the development of services that use these profiles. The second is focused on semantic spatial representation and the development of path finding and navigation techniques. Finally, the third area, which is the current thesis subject, is occupied with the user navigation tracking.

After the implementation of each of the three research parts, the most important and challenging task is to unify them and create the integrated Contextual Guide. The complete application will exploit all the provided functionality of the three related components by establishing a proper communication and information flow between them.

Especially, for the user tracking system, there could be some additional functionality, based on the collaboration with the user profile component. For example, the tracking system could notify the integrated application for a user who deviates too often and to suggest registering his/her profile information again. This action could detect user mistakes while filling the form of the profile manager, which could have led the system to propose not an optimal path.

Moreover, the tracking system could figure out, from user's deviations, some very interesting conclusions for his/her preferences. For example, if the application has suggested a route on the map which includes a stairway, while user deviated from the proposed route using an elevator, the system could conclude that he/she prefers to user motor passages. So, the path-finding component will take this parameter into consideration for the next path suggestion.

An issue of great importance for future investigation is the system expansion in multi-user environments. Specifically, the original intension is an integrated application which will support simultaneous tracking of all users who navigate with the aid of the Contextual Guide. The system architecture, as presented in previous chapters, is designed in order to handle such an extension, since the ontology-based architecture provides the ability for many user instances. As a consequence, the future investigation will include only the implementation characteristics which will enhance the indoor navigator with the potential of supporting interaction of more than one user at the same time.

The main implementation characteristic is the creation of a multithreaded application which will satisfy the needs for tracking many users simultaneously. This state obliges the existence of an appropriate control mechanism that will handle all the advanced operations and interaction modules. An example of such an advanced operation is the user feedback. In more detail, when a user reports an obstacle (e.g. a locked door), the system should examine the proposed paths of all the active users ensuring they will not confront the same obstacle during navigation. If the system detects paths containing the obstacle point, it dynamically re-plans them and informs users about the upcoming obstacle.

Every user instance will be handled by a specific thread and stored in a separate session. For crucial changes which influence all the users (e.g. user feedback) the ontology updates will be loaded in a local database. With such a design confrontation, the system will secure valuable navigation information related to the multi-user interoperability from possible system crashes during execution.

Since the Contextual Guide will be integrated, the evaluation testing will be more resourceful than the distinct efficiency examination of the three components. The testing results would be very useful in order to succeed an appropriate interaction for the user without inconveniencies. Finally, latency is a very crucial factor for the desired functionality and it should be reduced with applied techniques, in case it overpasses certain delay limits which will form an unpleasant user interaction.

Bibliography

[1]. J. Y. Yen, Finding the k shortest loopless paths in a network. Management Science, 17:712-716, 1971.

[2]. C. L. Forgy, Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem", Artificial Intelligence 19, 17-37, 1982.

[3]. G.D. Abowd, et al. Cyberguide: A mobile context-aware tour guide. Baltzer/ACM Wireless Networks, 3 (5): 421-433, 1997.

[4]. T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web”, Scientific American, May 2001.

[5]. M. Beigl, P. Gray and D. Salber, “Location Modeling for Ubiquitous Computing”, Workshop Proceedings, Ubicomp Atlanta, September 2001.

[6]. Y. Isakowski, F. Platz, A. Neumann and ETH Hoenggerberg, “Interactive Topographic Web-Maps Using SVG”, Open Zurich, 2002.

[7]. M. Weiser, “The computer for the twenty-first century” Scientific American, September 1991. (Reprinted in IEEE Pervasive Computing, Jan-Mar 2002).

[8]. G. Antoniou, F. Harmelen,”A Semantic Web Primer”, The MIT Press Cambridge, Massachusetts London England, 2003.

[9]. G. Antoniou, F. Harmelen, “Web Ontology Language: OWL”, April 2003.

[10]. H. Knublauch, “An AI tool for the real world Knowledge modeling with Protégé”, JavaWorld.com, June 2003.

[11]. R. Setchi, N. Lagos, “Ontology Development And Merging Using Protégé”, Cardiff School of Engineering, Cardiff University, UK, 2003.

[12]. J. Hightower, “From Position to Place”, Proceedings of the Workshop on Location Aware Computing (Ubicomp 2003), Seattle USA, October 2003.

[13]. H. Hu, D.L. Lee, Semantic Location Modeling for Location Navigation in Mobile Environment, proc. IEEE MDM’04, 2004.

[14]. C. H. Finin, T. Anupam, J. "An Ontology for Context-Aware Pervasive Computing Environments", University of Maryland Baltimore County, 2004.

[15]. G. Gartner, A. Frank, G. Retscher Pedestrian Navigation System in Mixed Indoor/Outdoor Environment- The NAVIO Project, CORP 2004 and Geomultimedia04, Vienna, Austria, 2004.

[16]. M. Baldauf, S. Dustdar, “A Survey on Context-aware systems”, International Journal of Ad Hoc and Ubiquitous Computing, 2004.

[17]. OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/, W3C Recommendation 10 February 2004.

[18]. OWL Web Ontology Language Reference, http://www.w3.org/TR/owl-ref/, W3C Recommendation 10 February 2004.

[19]. M. N. Sadeh, L. F. Gandon and B. O. Kwon: Ambient Intelligence: The MyCampus Experience, Technical Report CMU-ISRI-05-123, School of Computer Science, Carnegie Mellon University, 2005.

[20]. V. Tsetsos, C. Anagnostopoulos, P. Kikiras, T. Hasiotis and S. Hadjiefthymiades, "A Human-centered Semantic Navigation System for Indoor Environments", University of Athens, 2005.

[21]. P. Kikiras, V. Tsetsos and S. Hadjiefthymiades, "Ontology-Based User Modeling for Pedestrian Navigation Systems ", University of Athens, 2005.

[22]. C. Anagnostopoulos, V. Tsetsos, P. Kikiras and S. Hadjiefthymiades, "OntoNav: A Semantic Indoor Navigation System", proceedings of the

1st Workshop on Semantics in Mobile Environments (SME), MDM'05, Cyprus, May 2005.

[23]. M. O'Connor, H. Knublauch, S. Tu, B. Grosof, M. Dean, W. Grosso and M. Musen, "Supporting Rule System Interoperability on the Semantic Web with SWRL", Stanford Medical Informatics, Stanford University School of Medicine, Stanford, September 2005.

[24]. Yun Gil Lee, Jin Won Choi and Il Ju Lee, "Location Modeling for Ubiquitous Computing Based on the Spatial Information Management System", Journal of Asian Architecture and Building Engineering, Vol. 5, No. 1 pp.105-111, 2006.

[25]. I. Roussaki, M. Strimpakou, Carsten Pils, N. Kalatzis and M. Anagnostou, "Hybrid context modeling: A location-based scheme using ontologies", Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06), 2006.

[26]. B. Paepen, J. Engelen, "Using a Walk Ontology for Capturing Language Independent Navigation Instructions", Proceedings ELPUB2006 Conference on Electronic Publishing, Bansko, Bulgaria, June 2006.

[27]. S. Loke, "Context-Aware Pervasive Systems: Architectures for a New Breed of Applications", AUERBACH, Dec 2006.

[28]. The Protégé OWL API, http://protege.stanford.edu/plugins/owl/api/ .

[29]. Racer: Renamed Abox and Concept Expression Reasoner, http://www.sts.tu-harburg.de/~r.f.moeller/racer/.

[30]. Scalable Vector Graphics (SVG), W3C Recommendation, http://www.w3.org/Graphics/SVG/.

Table of Initials

Initials	Definitions
XML	eXtensible Markup Language
RDF	Resource Description Framework
OWL	Ontology Web Language
OWL DL	OWL Description Logic
API	Application Program Interface
UNA	Unique Name Assumption
DIG	DL Implementation Group
RACER	Renamed Abox and Concept Expression Reasoner
SWRL	Semantic Web Rule Language
LHS	Left Hand Side
RHS	Right Hand Side
SVG	Scalable Vector Graphics
WWW	World Wide Web
POI	Point of Interest
GUI	Graphical User Interface
URI	Unique resource identifier
UTO	User Tracking Ontology
JVM	Java Virtual Machine

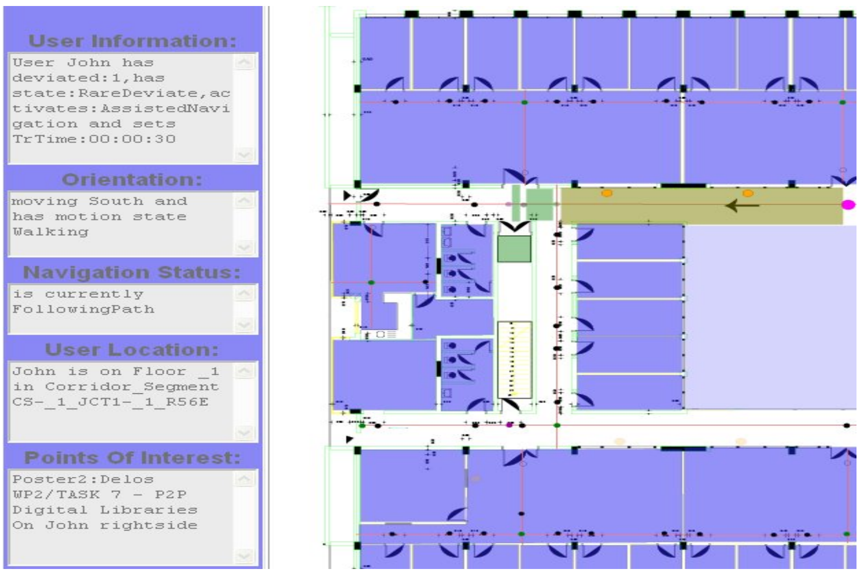
APPENDIX A:

Application Scenario Presentation

Scenario B

In the selected scenario, John is on the first floor of ICS-FORTH, at room exit R56E, which he reached after the end of scenario A in chapter IV. This time he wants to return to his original starting point, at the ground floor. After selecting start and destination points, the system highlights the new path.

Figure 1 shows the system's condition after the first click, which is updated in both the information and the map frame. In the map frame, system highlights the corridor segment in which John is and draws an arrow that informs for both exact position and direction. In the information frame, the Orientation, Navigation Status and User Location boxes have been updated. Orientation box informs John that he is moving South and his motion state is Walking. His Navigation Status is FollowingPath as John walks inside a highlighted area which belongs to the suggested route. The User Location box informs John that he is on first floor (Floor_1) and in corridor segment CS-_1_JCT1-_1_R56E. The Points of Interest box has been updated and presents data about the highlighted POI that John has selected. Moreover, John is informed about the location relation between him and the selected POI, which states that the POI is on John's right side.



```
User John has
deviated:1,has
state:RareDeviate,ac
tivates:AssistedNavi
gation and sets
TrTime:00:00:30
```

```
moving South and
has motion state
Walking
```

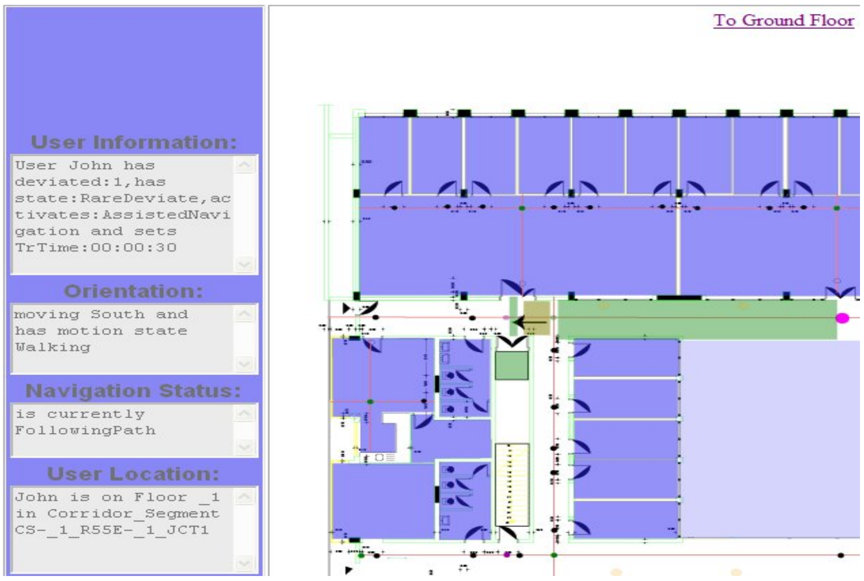
```
is currently
FollowingPath
```

```
John is on Floor _1
in Corridor_Segment
CS-_1_JCT1-_1_R56E
```

```
Poster2:Delos
WP2/TASK 7 - P2P
Digital Libraries
On John rightside
```

Figure 1: User starts navigation on the proposed path

Figure 2 shows the system's condition as John continues his navigation and reaches the elevator. In the map frame, system highlights the corridor segment in which John is and draws an arrow that informs for both exact position and direction. In the information frame the User Location box has been updated, as it informs John that he is on first floor (Floor_1) and in corridor segment CS-_1_ R55E-_1_JCT1. However, before John enters the elevator, he realizes that it is not working, so he decides to report it to the system by clicking on the 'Report Obstacle' button and on the elevator area on the map.



User Information:

```
User John has
deviated:1,has
state:RareDeviate,ac
tivates:AssistedNavi
gation and sets
TrTime:00:00:30
```

Orientation:

```
moving South and
has motion state
Walking
```

Navigation Status:

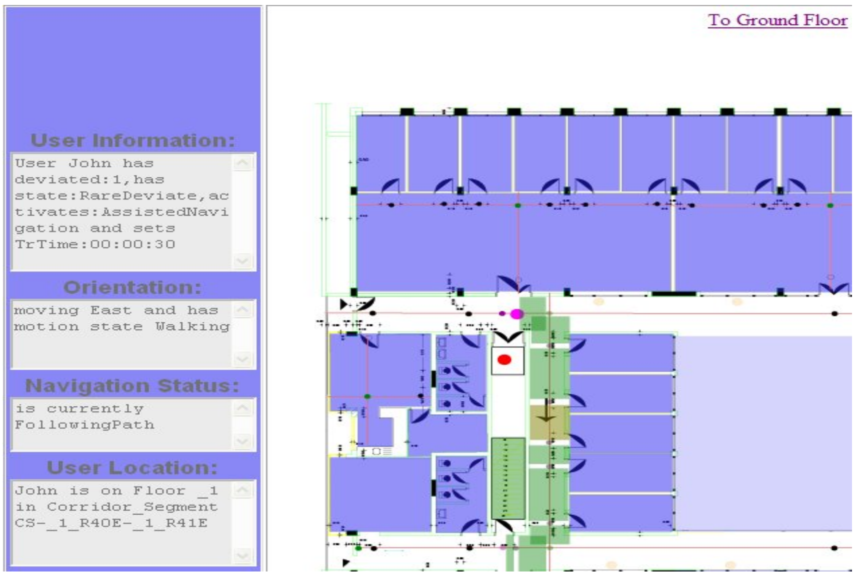
```
is currently
FollowingPath
```

User Location:

```
John is on Floor _1
in Corridor_Segment
CS-_1_R55E-_1_JCT1
```

Figure 2: User continues navigation and reaches the elevator

System receives the feedback from John and dynamically triggers re-planning process and highlights on John's map the new optimal path to his desired destination, as shown in Figure 3. The red dot on the elevator indicates the malfunction, whereas the purple dot shows the new starting point of John's navigation. In the information frame, the Orientation and User Location boxes have been updated. Orientation box informs John that he is moving East and his motion state is Walking, whereas the User Location box informs that he is in a new corridor segment (CS-_1_ R40E -_1_R41E).


User Information:

```
User John has
deviated:1,has
state:RareDeviate,ac
tivates:AssistedNavi
gation and sets
TrTime:00:00:30
```

Orientation:

```
moving East and has
motion state Walking
```

Navigation Status:

```
is currently
FollowingPath
```

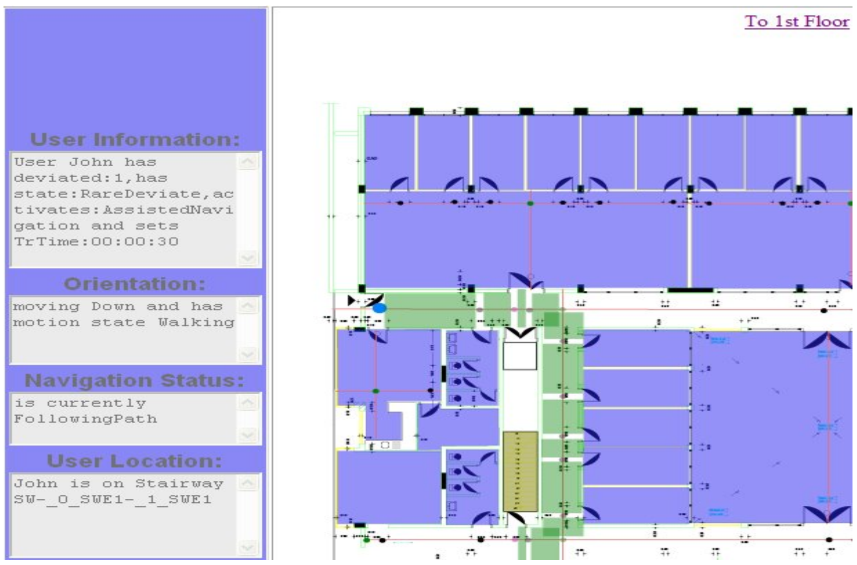
User Location:

```
John is on Floor _1
in Corridor_Segment
CS-_1_R40E-_1_R41E
```

Figure 3: User reported elevator malfunction and the system re-planned the path

In Figure 4, John is on stairs (the light green box) and gets down to the ground floor of ICS-Forth. The ground floor's proposed path is highlighted until the destination point, which is depicted by the blue dot. The information frame is updated. The Orientation box shows new moving direction for John, the down direction, as the stairway leads him from an upper floor to a lower floor. User Information and Navigation Status boxes have not changed as user is still following the proposed path. Finally the User Location box has also changed, as John is informed about being on a stairway SW-_0_SWE1-_1_SWE1 (i.e. the stairway which connects ground floor stairway exit SWE1 with first floor exit). Moreover, floor relation between user and floor space element has been deleted, supposing that the stairway does not belong to a specific floor.

In the last scenario screenshot, Figure 5, John continues his navigation following the highlighted path on the ground floor. The black arrow indicates his location and direction towards the destination point (the blue dot). The Orientation box informs John that he is moving West according to the magnetic compass and has Walking motion state. The User Location box states that John now walks inside corridor segment CS-_0_ R40E -_0_R41E and navigates on the ground floor of the building (Floor_0).



User Information:

User John has deviated:1,has state:RareDeviate,activates:AssistedNavigation and sets TrTime:00:00:30

Orientation:

moving Down and has motion state Walking

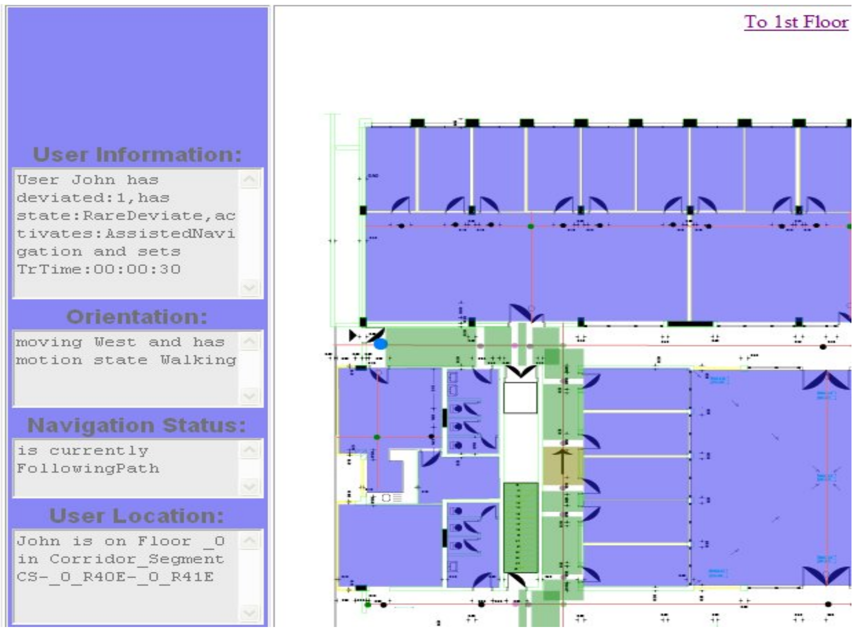
Navigation Status:

is currently FollowingPath

User Location:

John is on Stairway SW-_0_SWE1-_1_SWE1

Figure 4: User takes the stairs and descends to the ground floor



User Information:

User John has deviated:1,has state:RareDeviate,activates:AssistedNavigation and sets TrTime:00:00:30

Orientation:

moving West and has motion state Walking

Navigation Status:

is currently FollowingPath

User Location:

John is on Floor _0 in Corridor_Segment CS-_0_R40E-_0_R41E

Figure 5: User continues the navigation with direction the destination point