VISIT COLLAGE: Generating Digital Journals to Prolong the Afterglow Effect of Visits in AmI Exhibition Spaces

Stavros Bastakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Knossou Av., P.O. Box 2208, Heraklion, GR-71409, Greece

Thesis Advisor: Prof. *Constantine Stephanidis*

This work has been supported by the Institute of Computer Science of the Foundation for Research and Technology-Hellas (FORTH).

UNIVERSITY OF CRETE

COMPUTER SCIENCE DEPARTMENT

VISIT COLLAGE: Generating Digital Journals to Prolong the Afterglow Effect of Visits in AmI Exhibition Spaces

Thesis submitted by
Bastakis Stavros

in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Bastakis Stavros

Committee approvals: _____
Constantine Stephanidis
Professor, Thesis Supervisor

_____
Dimitris Plexousakis
Professor, Committee Member

_____
Dimitris Grammenos
Principal Researcher, Committee Member

Department approval: _____
Angelos Bilas
Professor, Director of Graduate Studies

Heraklion, November 2013

VISIT COLLAGE: Generating digital journals to prolong the afterglow effect of visits in AmI exhibition spaces

Stavros Bastakis

Master's Thesis

Computer Science Department, University of Crete

Abstract

In the last decade, the emerging domain of Ambient Intelligence influenced the introduction of innovative, even experimental, ubiquitous technologies in exhibition spaces. Exhibition spaces, and in particular museums, are constantly augmenting their spaces with digital installations, thus the necessity of systems that deliver context-sensitive information and tailored recommendations in order to create new and engaging forms of interactive experiences is even higher.

The VisitCollage system proposed in this thesis aims to enhance the visitors' experience and engagement in the ubiquitous exhibition space. The system provides facilities for capturing the visitors' activity within the exhibition space in real time, as well as mechanisms for identifying visitors' interests and behavior based on their activity. In addition, it supports the delivery of personalized recommendations of related exhibits, either physical or digital, based on the automatically inferred interests. The visitors' interests along with the recommendations are visualized into personalized, dynamically generated web pages, thus conveying the on-site exhibition experience with digital content on-line.

In order to achieve the above, the VisitCollage system provides an infrastructure of intelligent agents which unobtrusively monitor the visitors' activity in a seamless manner, and exploiting the advantages of ontology-based modelling and rules-based reasoning, infers the visitors' interests and recommendations. Advanced

web technologies such as HTML5, CSS3 and JavaScript are utilized to “capture” the entire visit into dynamically generated, personalized journals where the captured content is visualized: (i) sequentially, ordered either chronologically or by interest, (ii) overlaid on top of a floor plan and (iii) as an elegant booklet.

VISIT COLLAGE: Ψηφιακά ημερολόγια εμπειριών χρήσης σε εκθεσιακούς χώρους Διάχυτης Νοημοσύνης

Σταύρος Μπαστάκης
Μεταπτυχιακή Εργασία
Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Περίληψη

Την τελευταία δεκαετία, ο αναδυόμενος τομέας της Διάχυτης Νοημοσύνης ενέπνευσε την εισαγωγή καινοτόμων, ακόμη και πειραματικών, τεχνολογιών Διάχυτης Νοημοσύνης στους εκθεσιακούς χώρους. Οι εκθεσιακοί χώροι, και συγκεκριμένα τα μουσεία, συνεχώς εμπλουτίζουν το χώρο τους με ψηφιακά διαδραστικά συστήματα, επομένως προκύπτει η ανάγκη για συστήματα που παρέχουν πληροφορίες και προτάσεις προσαρμοσμένες στο εκάστοτε πλαίσιο χρήσης, με σκοπό τη δημιουργία νέων και ελκυστικών μορφών διαδραστικής εμπειρίας.

Το σύστημα VisitCollage που προτείνεται στη παρούσα διατριβή στοχεύει στην βελτίωση της εμπειρίας χρήσης των επισκεπτών σε τεχνολογικά επαυξημένους εκθεσιακούς χώρους. Το σύστημα καταγράφει τη δραστηριότητα των επισκεπτών μέσα στον εκθεσιακό χώρο σε πραγματικό χρόνο, και με κατάλληλους μηχανισμούς εξάγει τα ενδιαφέροντα των επισκεπτών βασιζόμενο στη συμπεριφορά τους. Τα ενδιαφέροντα και οι προτάσεις οπτικοποιούνται μέσω δυναμικά παραγόμενων ιστοσελίδων, αποτυπώνοντας με αυτό το τρόπο την πραγματική εμπειρία στον εκθεσιακό χώρο σε διαδικτυακό ψηφιακό περιεχόμενο.

Για την επίτευξη των παραπάνω, το σύστημα VisitCollage παρέχει κατάλληλες υποδομές έξυπνων πρακτόρων οι οποίοι παρακολουθούν τη δραστηριότητα των επισκεπτών με ένα διακριτικό και αδιάλειπτο τρόπο, και αξιοποιώντας την αντίστοιχη μοντελοποίηση βάση οντολογιών και την συλλογιστική βάση κανόνων, παράγει συμπεράσματα για τα ενδιαφέροντα των επισκεπτών καθώς και πιθανές

προτάσεις. Προηγμένες τεχνολογίες διαδικτύου, όπως HTML5, CSS3 και JavaScript αξιοποιήθηκαν για να αποτυπωθεί το σύνολο της επίσκεψη σε δυναμικώς παραγόμενα, προσωποποιημένα ημερολόγια όπου το συλλεχθέν περιεχόμενο οπτικοποιείται: (i) σειριακά, ταξινομημένο είτε χρονολογικά είτε με βάση το ενδιαφέρον, (ii) επαυξάνοντας μια σχεδιαστική κάτοψη του χώρου και (iii) σαν ένα κομψό βιβλίο επίσκεψης.

Ευχαριστίες (Acknowledgements)

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον επόπτη της μεταπτυχιακής μου εργασίας και καθηγητή μου, κ. Κωνσταντίνο Στεφανίδη, για την καθοδήγηση και την υποστήριξη που μου προσέφερε στο πλαίσιο της συνεργασίας μου με το Εργαστήριο Ανθρώπου-Υπολογιστή του Ινστιτούτου Πληροφορικής(ΙΠ) του Ιδρύματος Τεχνολογίας και Έρευνας(ΙΤΕ). Επίσης, ευχαριστώ ιδιαίτερα τους κ. Δημήτρη Πλεξουσάκη και κ. Δημήτρη Γραμμένο για τη συμμετοχή τους στην εξεταστική επιτροπή και στην αξιολόγηση της παρούσας εργασίας.

Στη συνέχεια θα ήθελα να ευχαριστήσω τη Μαργαρίτα Αντόνα, τη Σταυρούλα Ντοά, την Ίλια Αδάμη, την Μαρία Κορόζη και τον Γιώργο Μαργέτη για την πολύτιμη βοήθεια και τις συμβουλές τους στις διάφορες φάσεις της εργασίας μου. Ένα μεγάλο ευχαριστώ οφείλω στον Αστέριο Λεωνίδη για την άψογη συνεργασία μας και τις ατελείωτες ώρες που περάσαμε μαζί σε συναντήσεις, κουβέντες, προτάσεις και διαφωνίες.

Επιπλέον, θα ήθελα να ευχαριστήσω όλους τους συναδέλφους και φίλους μου για τη συμπαράσταση και τη στήριξη τους καθ' όλη τη διάρκεια της εκπόνησης της μεταπτυχιακής μου εργασίας. Ιδιαίτερα θα ήθελα να ευχαριστήσω τους Αντώνη Κ. , Χρήστο Α. , Γιώργο Μ. , Γιάννη Δ. , Χρύσα Μ. , Γιώργο Γ. , καθώς επίσης τους Σήφη Σ. , Χάρη Μ. , Γιώργο Σ. , Βαγγέλη Χ. , Μηνά Σ. , Αλώρα Σ. και Ειρήνη Μ.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τη Βαλεντίνα, η οποία εκτός από την αμέριστη υπομονή και συμπαράσταση της σε όλες τις φάσεις της παρούσας εργασίας, βρισκόταν δίπλα μου σε κάθε στιγμή δίνοντας μου την απαραίτητη ώθηση να κυνηγάω τις προσδοκίες μου.

Τέλος, το μεγαλύτερο ευχαριστώ το οφείλω στους γονείς μου Νίκο και Κική καθώς και στα αδέλφια μου Γιάννη και Νίκο, που ήταν πάντα δίπλα μου εμψυχώνοντας με σε κάθε φάση της ζωής μου. Η αγάπη τους και η στήριξη τους αποτέλεσαν για μένα πηγή για να αγωνίζομαι και να εκπληρώνω τους στόχους μου. Ελπίζω αυτή η εργασία να αποτελεί μια μικρή «ανταμοιβή» για τις θυσίες που έκαναν για μένα.

To my parents

Nikos and Kiki

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 Ambient Intelligence

The continuous evolution of Information Technology since the introduction of the first computer systems has drastically affected the way people interact with computers and electronic devices. Nowadays, more and more people enjoy access to information anytime anywhere. This evolution has increased the users' expectations for innovative technologies, thus expanding the potential of technology to be integrated in the natural environment through the development of novel concepts and tools that provide content-rich invisible computing.

Ambient Intelligence (AmI) [1] refers to environments that are sensitive and responsive to the presence of people and reflects a vision on the future of computing, electronics and telecommunications. Ambient Intelligence envisions the encapsulation of technology in our natural surroundings, so that it is present when we need it, context-sensitive and autonomous. Ambient Intelligence offers great opportunities to enrich everyday activities and dramatically change the way of life by using the information and intelligence incorporated in a network of "smart" devices hidden into our surroundings. As these devices grow smaller, more connected and more integrated into our environment, the technology disappears in the background and only the user interface provided remains perceivable.

AmI combines Ubiquitous Computing [2] with Intelligent User Interfaces [3] in order to offer the users natural means of interaction. Ubiquitous computing includes the monitoring of the environment, making the system aware of the user's location and goals. The input of the system may be acquired through sensors, cameras and any device that can extract information about the user's movement and interaction with the environment. The user's communication with the system is transformed in such a way so that users have the impression of interacting with the environment rather than a personal computer or any electronic device.

AmI addresses various everyday living environments and activities including, but not limited to, home, healthcare, learning and education, entertainment, work, arts and culture. All these applications share, as a common feature, the purpose of responding to the users' needs and preferences. This thesis investigates the potential of AmI in exhibition spaces, and specifically the way AmI can assist in enhancing visitors' experience and engagement.

1.2 Motivation for capturing and presenting users' visit experience

In the last decade, progress in multimedia has allowed for new, experimental forms of communication through the use of ubiquitous technologies in exhibition spaces. Clearly, the focus of the digital exhibition spaces should be on the visitor's experience rather than the technology itself. However, such a goal cannot be achieved without the aid of advanced technology. This thesis focuses on the aspect of user experience and introduces strategies that aim to enhance visitors' engagement within an exhibition space.

Exhibition spaces and especially museums are increasingly augmenting their environments with digital installations and systems that deliver context-based information and recommendations in order to create new and engaging forms of interactive and collaborative experiences. Personalization [4] is the fundamental process by which engagement can be achieved. Towards this objective, systems should be able to recognize the users' interests and behavior in order to deliver content suitable to their profile. Semantic Web approaches are regarded as crucial in this procedure in order to overcome the semantic gap between the diverse digital content and the users' preferences. Moreover, the Semantic Web in the exhibition context offers a way to analyze and understand humans' behavior in order to provide recommendations based on the users' interest and the semantically annotated context. Last, in order to strengthen the experience of the visitors, the visualization of the personalized content should be taken into account in the development of such systems. Visualization [5] is the keystone of user

experience, as it constitutes the means by which the captured experience is presented to the visitor.

The VisitCollage system presented in this thesis, aims to exploit all the aforementioned aspects in order to enhance visitors' experience in a ubiquitous exhibition space. As regards personalization, the proposed system recognizes the users' interests and behavior based on their interaction within the exhibition space in a transparent manner, and uses ontologies and semantic rules to discover and deliver personalized recommendations for related content, exhibits and places of interest. Finally, VisitCollage reconstructs the captured experience into personalized, dynamically generated web pages, which the visitor can later explore and share through social networks.

1.3 Thesis structure

The rest of this thesis is structured as follows:

- Chapter 2 presents an overview of related work and the contribution of the VisitCollage system.
- Chapter 3 describes the requirements elicitation process.
- Chapter 4 provides an overview of the underlying system describing the main concepts behind its development.
- Chapter 5 describes in-depth the overall architecture and provides technical details regarding the implementation of each sub-system.
- Chapter 6 presents the expert-based heuristic evaluation process of the system and examines its findings.
- Chapter 7 summarizes the conclusions of this thesis and outlines the future steps.

Chapter 2
Related Work

In the past few years, intelligent exhibition systems have become an interesting topic that has attracted many researchers' interest. Several systems have been proposed for personalizing visitors' experiences by capturing interests while on-site or before their visit online. Other systems take a step further by proposing personalized recommendations tailored to the visitors' interests.

2.1 Towards Personalized Visits in Exhibition Spaces

Hitzeman et al in [6] describe a system called ILEX that aims to generate descriptions of objects displayed in a museum gallery. The ILEX's domain is that of a 20th Century Jewellery Exhibit in the Royal Museum of Scotland. ILEX is a system designed to create captions (labels) for dynamically generated web pages containing text and graphics for an online museum gallery. The labels are tailored to each user based on their experience, interests, and the course of their visit to the gallery. ILEX stores a history of what labels a user has seen in order to maintain a model of the user's knowledge and uses it to make its presentation less repetitive and more context-sensitive.

The mEXPRESS project presented in [7] aims to exploit the technological opportunities arising from evolution in the areas of wireless networks and indoor positioning technologies, such as WLAN and Indoor-GPS in order to support and facilitate the professional exhibition industry in a context-aware manner. The most important services for visitors include navigation planning, on-route bookmarking, provision of content for nearby exhibits and notifications. The terminal devices that can be used to access the mEXPRESS services include: (i) PDAs that are used to mark bookmarks, receive multimedia content, notifications and routing information relative the location of the visitor. (ii) Mobile-phones (or smart-phones), (iii) PCs used by the visitors before or after the exhibition and (iv) info-

kiosks used by the visitors at the exhibition to review the collected material or view their virtual trail within the exhibition.

The PEACH project [8] [9] develops a PDA-based location-aware museum tour application whose content is adapted to each visitor. The system is initialized with a user profile and then, in the course of the visit, adapts to the behavior of the visitor, proposing personalized, context-dependent presentations. Both mobile and stationary devices are exploited, and a virtual presentation agent transfers from the handheld device to the Virtual Windows (large displays) when such devices are in sight, and assists the visitor. Regarding identification, artificial vision-based technology pinpoints what region the visitor is currently looking at, so that relevant presentations about the details of that area in the focus of the visitor's attention can be provided. Moreover, in the course of a visit, the system capitalizes on each individual visitor's feedback whenever possible to guarantee appropriate presentations for their interests and preferences. Adaptivity is achieved by considering the elements that seemed of major interest to the visitor and tailored to multimedia presentations. At the end of the visit, a personalized written report that summarizes the key aspects of the overall visit experience is automatically produced for the visitor to take home or to get as an electronic diary. One of the claimed principles of the system is the unobtrusiveness of the visit support: the system should never intrude between the exhibit and the visitors, letting them focus their attention on real things.

At the Exploratorium [10] [11] [12], a science museum in San Francisco, an RFID-based application called eXspot is developed to augment visitors' experience. In particular, visitors wear RFID-enabled cards or carry PDAs [13] through which they interact with networked RFID readers or infrared transmitters in the case of PDAs, which are mounted on museum exhibits. Through the portable PDAs, they can view more content about the exhibit in the embedded web browser. The exhibits are hands-on experiments which visitors take part in (for example, a heat camera that shows a thermal image of the body). As the visitor takes part in an exhibit experiment, their RFID tag is read and triggers a camera to record the experiments or take digital images of them. The results of these experiments are

automatically uploaded to a personalized webpage for the visitor, which provides a record of their visit including the exhibits visited and the photographs taken. Moreover, the webpage provides suggested links to additional online content and teaching materials related to the exhibits.

In [14] Yu et al propose a scalable context-aware intelligent museum system called iMuseum. The system can capture the information of the visitor and surroundings, recognize the visitor's purpose and assist visiting in the museum. The system is based on a context model that integrates both an ontology-based model and a database in which data are maintained and retrieved. The ontology-model is used in terms of reasoning on domain knowledge. Regarding the implementation, the context server, the context-aware applications and other basic services are implemented on a background computer. Every visitor is given a PDA to be used as an assistant tour tool in which customized relic information is presented. The PDA connects with the background computer trough WLAN Access Point and is equipped with an RFID reader that can detect tags attached on the relics. The interest of the visitors is inferred from the time localized near a relic and once they approach the PDA to the RFID tag attached to the relic the system decides whether the visitor is interesting or not in this relic. If interested, the corresponding multimedia commentaries of the specific relic are delivered to the visitor's PDA. There are different modal commentaries available for each relic, which can be used according to context. For example, in case of slow network speed, text descriptions are provided and in case of younger users, images are presented. Video commentary is also supported, but requires broader network bandwidth.

The Experience Re-player (ER) presented in [15] is a system that utilizes smart-phones to record experiences of a visit, which are augmented with additional multi-media content and reconstructed digitally in the form of an interactive website. The ER follows the concept of Remember [13] that allows for richer interactions within the environment, integrates external multimedia content and enables editing of the experience record. The ER system exploits three main aspects of smart-phones to enable pervasive computing: their auto-identification

capability through their wireless interface or their secondary wireless local networking capability via Bluetooth, their integrated camera that allows for the capture of video or images and last, their ability to record and transmit voice messages. The system consists of four main components: (i) the Experience Recorder, which collects the visitor's experiences, captured by their smart-phones and associates them with their identity; (ii) the Navigation Engine, which is at the core of the ER system and is the component that reconstructs the captured experiences into trails; (iii) the Experience Re-player which reconstructs the experience as a web application that combines spatial visualizations and multimedia content; and finally, (iv) the Visitor Analyst component which is used to aggregate and analyze recorded trails in order to gain a better understanding about common behaviors among the visitors.

In [16] the authors propose a digital storytelling project for museums titled CHESS. The principal objective of the proposed framework is to enable both the experiencing of personalized interactive stories for the visitors and the authoring of narrative structures by the cultural content experts. The proposed storytelling model for CHESS attempts to capture and structure the knowledge behind the way humans are creating the stories and eventually how these are presented to the visitors. Towards this direction, five major phases are identified. First, the scripting phase, during which the basic archaeological knowledge is transformed into emerging stories. The second is the staging phase where the stories created are associated with exhibits, paths and hotspots in the physical space of the exhibition. Next, the producing phase takes place where digital content is created and imported into the CHESS in order to be available for the editing phase where the digital resources are appropriately arranged to create the final complete story. An Authoring Tool is designed and implemented to support this phase. Finally, in the experiencing phase, the authored stories are adaptively provided to visitors through web-based and mobile systems.

2.2 Personalized Recommendations in Exhibition Spaces

Liiv et al in [17] describe a platform for recommendations in the cultural domain called SMARTMUSEUM. In this project, they aim to improve personalized on-site cultural heritage access with focus on user preferences and recommendations. SMARTMUSEUM uses a combined approach based on rules, collaboration and content personalization, where content is semantically annotated using an ontology. Recommendations are about cultural objects allocated in the museum and content related with those objects. User profile includes abilities and interests of user, and a log of visited places. In particular, the visitors' impressions regarding physical artifacts are monitored using various technologies such as eye movement monitoring, and RFID tags accessible with portable readers. Moreover, GPS technology is used for recognizing objects outside the museum. The visitor is provided with a PDA, which is the main device for both determining their location and presenting information. Additionally, the visitors can manually mark an object in the PDA as "I like it" or "I do not like it" or add short comments about items visited. In order to specify interest, the system also measures the approximate time of looking at the object and browsing the available information about it. When visitors leave the museum, they can view their profile and interesting items from their visit through a web application from an ordinary web browser. Although being an interesting system, SMARTMUSEUM is mainly targeted to physical exhibits exploration and requires handheld devices like PDAs from the visitors in order to specify their location and present content.

Based on the experiences of the SMARTMUSEUM project, the authors in [18] present the architecture of a personalized recommendation system using probabilistic reasoning. The system creates plans for visiting interesting objects and events on the trip, based on the personal preference profile. The SMARTCITY project is a semantic recommender and route composer system for tourists. Based on the user's location, time of visit and preferences, its suggestion engine finds interesting objects for the given user. Each object is assigned a score of interest thus the objects that the user probably likes achieve a higher score. The objects are

organized into a timetable based on their location and time and finally the schedule and the trip route are presented to the user. The visitors also have the option to modify the suggested objects in order to create a more suitable timetable for them.

The Digital Backpacking project [19] at the Austrian Museum of Techniques proposes an object selection strategy based on an RFID smartcard. The exhibition combines traditional object exhibits, computer-enhanced hands-on exhibits, and a large space dedicated to modern media. As a central element of this exhibition, visitors can buy a smartcard that enables them to store collected or self-created data in a 'digital backpack', which can be either viewed in loco on terminals or to kept as a souvenir for post-visit debriefing. Card owners interact with the information system of the exhibition that is accessible at diverse installations and terminals. Digital objects created by the visitor such as news, images, videos, sound samples or game results are stored in their profile along with objects from other exhibits the visitor interacted with. Once the card is laid on the reader area of guide systems, the visitor can explore their profile or stored content, communicate with other visitors within the exhibition or even read recommendations on what to visit next based on others visitors' path compared with theirs. Moreover, the system includes a public backpack where the visitor can see the last ten videos, images, sound samples that were produced within the exhibition. Visitors can also access their digital backpack via the museum homepage, by using a number printed onto their smartcard as login, and download the content as a personal souvenir.

Authors in [20] present a relevance-feedback based preference learning mechanism for multimedia personalization in a pervasive environment, where the high-capability master device learns the user's preference by compiling statistical analysis based on the feedback from the low-capability slave devices, which observe the user's behavior. The master device supports both explicit input/modification and implicit learning, which is achieved through an algorithm that applies relevance feedback exploiting the Naïve Bayes classifier.

The museum experience described in [21] is a very good example of the use of semantic descriptions in a real-time application. The augmented audio reality system named e(c)ho uses inference rules along-side representation of user models, content descriptions and involves several ontologies including the CIDOC CRM ontology for modeling museum artifacts and the ontologies developed for sound objects and exhibition (space). The ec(h)o project designed a platform to create a museum experience that consists of a physical installation and an interactive virtual layer of three-dimensional soundscapes that are physically mapped to the museum displays. The retrieval mechanism is based on the user model and conceptual descriptions of sound objects and museum artifacts. The interface of ec(h)o does not rely on portable computing devices; rather it utilizes a combination of gesture and object manipulation recognized by a vision system. An enhanced experience for the museum visitors is achieved without inserting an extra layer of technology between the visitor and the museum exhibit. In ec(h)o, each artifact/exhibition is annotated with a set of concepts and user interests are represented as a set of weighted concepts from the ontology and are inferred from the user's movement through the exhibition as well as from the visitor's interaction with the sound objects. Based on the model of interests, ec(h)o recommends to a user the next best audio fragment. The rule-based user model was specifically designed to work in environments where rich semantic descriptions are available. The retrieval criteria are represented as inference rules that combine knowledge from psychoacoustics and cognitive domains with compositional aspects of interaction. Overall, ec(h)o proved that ontologies and rules provide an excellent platform for building a highly-responsive context-aware interactive application.

In [22], [23], [24] and [25] the authors experimented with semantic web technologies to enrich the presentation of the Rijksmuseum collection with information retrieved from public ontologies. The CHIP demonstrator was developed to demonstrate how Semantic Web and personalization technologies could be deployed to enhance access to digital collections of museums. The CHIP demonstrator works with the Amsterdam Rijksmuseum ARIA database, which

contains images and descriptions and consists of three main components: (i) The *Art Recommender*, a web-based component realized as Java Servlets, JSP pages, CSS and JavaScript. This component helps users, discover their art interests in the museum collection by rating the existing artifacts in a 5-degree rating scale. Based on the ratings the visitors' profile is updated with a list of recommended artworks. (ii) The *Tour Wizard* component is also web-based and according to the collected ratings generates online museum tours, which can be presented both on a geographical museum map and in a historical timeline. (iii) The *Mobile Guide* component converts online museum tours generated from the *Tour Wizard* to the on-site tours on a PDA, and assists the user to find their way during the visit. Rating of artworks is also supported during tour with RFID tags through the reader attached to the PDA, which in parallel presents information about the artwork. Once the tour is finished, the device sends the user's behavior to update the user model on a Web Server. In order to address the cold-start problem, where the visitors provide limited input to the *Art Recommender* component to generate recommendations, the users' tags from iCity [26] are mapped to CHIP art topics. This enables the population of the user model in CHIP and instant generation of recommendations.

The REMIX project proposed in [27] aims to develop a personalization platform for museums based on RFID technology and advanced recommender-systems algorithms. Two major cases are examined: the online and the offline case. In the online case, based on the visitors' movements, which are tracked non-intrusively by RFID sensors placed on the exhibits, the visitors are able to get recommendations for exhibits during their visit. In the offline case, on the other hand, once the visitors leave the museum, they can connect to a personalized web-based application, which provides information about their actions during the visit, about the exhibits that were interesting for them and see recommendations for potentially interesting exhibits that they can see in future visits. In addition, the tracked information can help the museum's management to understand the behavior and preferences of the visitors and shape evaluation metrics. Particularly, regarding the recognition of the visitors, RFID tags are carried by the visitors

which can be obtained at a kiosk at the museum's entrance and may perform simple registration providing low privacy data such as their email address, favorite color etc. ,which increase the security in case the RFID tag is lost. The visitors can bookmark an exhibit, thus recording it as visited, by swiping their RFID tag at the vicinity of the corresponding RFID reader. The RFID reader package reads the RFID tag of the visitor and sends the ID of the tag to the base station, via the wireless network, along with the ID of the exhibit and the current time. The database contained in the base station maintains information about the exhibits, the visitors and data about the interaction of the visitors with the exhibits following the principles of relational database development. The information stored in the database of the base station are used to create personalized Web pages for each visitor who can logon by using the ID of the RFID tag as user-name and as password other information provided during registration such as the email. In the personalized Web pages, the user can view information such as the date or hour the exhibits were visited and additional teaching material, like links to online encyclopedia articles related to the visited exhibits. The Web pages also accommodate the recommendations of exhibits that can be viewed in future visits. Finally, the provided services of REMIX to the museum's management include both mining sequential patterns, such as sequences of interactions that tend to appear more frequently, and mining temporal patterns such as the discovery of trends or discovery of the "life-cycle" of new exhibits.

Meehan et al in [28] and [29] propose the VISIT system that is based on a hybrid recommendations approach made up of collaborative filtering, content-based recommendations and demographic profiling. Reasoning is performed as part of the hybrid system to determine the weight/importance of each different context type. The system takes into consideration five main types of contextual data: location, time, weather, social media sentiment and personalization. The location data are sensed through GPS, GSM and Wi-Fi and used in order to provide the visitor with information about the place they are visiting and suggest a list of points of interest for that place. The time data are used to allow the application to determine if the POI is open before suggesting it to the user, and to calculate the

amount of time the visitor stays at each attraction, thus determining the level of interest in a particular attraction. The WorldWeatherOnline API provides the weather data by which the system determines whether the prevailing conditions are favorable for visiting outdoor attractions or prepare the user to travel or not depending on the weather conditions. Moreover, the system performs sentiment analysis on twitter messages in real time to determine the current "mood" of each tourist attraction. This is calculated as percentage of positive, negative and neutral tweets about a point of interest. Finally, regarding personalization, the system retrieves personal data stored within a social network such as Facebook with the approval of the user in order to ensure that the suggestions are as relevant as possible to the current user. The main types of data collected from the social network include the age, gender, relationship status, number of children, which are used as a starting point for the application and can be changed explicitly by the user in the application. The system continues to implicitly learn from user behavior and their profile is updated incrementally. The system is built on the android platform and the initial data are imported using XML and maintained in an SQL server database on a web server. Last, the decision-making mechanism utilizes artificial neural networks, fuzzy logic and principal component analysis techniques.

2.3 VISIT COLLAGE: Contribution

As the interest for intelligent exhibition spaces grows, the approaches towards this issue expand covering a wide range of topics. In the last decade, the progress in multimedia has allowed for new, experimental forms of communication by the use of ubiquitous technologies, thus exhibition spaces are increasingly augmenting their environment with digital installations. However, attention is mostly focused on the digital exhibition spaces and the technology itself rather than on the visitor's experience. The approach proposed in this thesis focuses on how to increase the visitors' engagement and improve their visit experience by transforming their interaction into personalized web content, which they can further explore.

The VisitCollage system proposed aims to cover the gap between the digital exhibition artifacts and the visitors' engagement in the exhibition spaces by providing mechanisms targeted to enhancing their experience. The system is separated from other applications running in the digital exhibition space, hence it communicates efficiently with them through services. It offers straightforward registration services for new users and aggregation services for heterogeneous data sources. Flexibility in the identification method is also enhanced, as the actions' services provided to monitor the visitors' activity only require a unique id regardless of the way the visitors' are identified within the exhibition.

Another major characteristic of VisitCollage is that it monitors the visitors' activity in an unobtrusive seamless manner. Monitoring is not based on external devices such as PDAs or smartphones, which implies further complexity for the users, rather the visitors' activity is captured through the digital exhibits the visitors interact with, based on mechanisms supplied by VisitCollage. Moreover, monitoring is not intended to capture specific moments of the users' visit; instead, the system monitors the whole interaction of the visitors' in order to afford a complete view of their experience within the exhibition space.

In terms of personalization, the visitors' interests are inferred both implicitly by observing users' interaction within the multimedia of the exhibits and explicitly as provided by the users when filling in their profile. Both implicit and explicit data is exploited in the process of interests' extraction and related content generation in order to recommend items relevant to users' interests. Towards this end, the system supports rule-based inferences as well as content-based and contextual recommendations based both on items' features and descriptions and on similarities in the user profile and the candidate items. In order to face the cold-start problem, which appears when there is a limited amount of items visited, the semantic relations of the tags are used in order to provide additional concepts for which related content will be searched.

Another major characteristic of VisitCollage is the visualization of the monitored information into dynamically generated and personalized web pages.

The web-based visit journals produced are unique for each visitor and tailored to their interaction among the smart exhibits presenting the data mining results, such as the interesting items and recommendations. The system supports the presentation of content in various forms: (i) Sequential, in which the content is presented vertically, sorted either chronologically or by the interest weight of the items in descending order, (ii) the spatial representation, in which the visited systems are represented as points on the floor plan of the exhibition while the interesting items for each system are presented below the floor plan once a system is selected along with their additional content and recommendations and finally, (iii) the booklet representation offers the possibility to the user to browse the content of their visit through a digital flipping book.

Finally, the content can be presented either on-site via a web browser provided by the exhibition at the end of the visit or off-site once the visitor leaves the exhibition. The presentation of the visit history and recommendations deepens visitors' experience beyond a single visit, increases their satisfaction, and motivates them to plan future visits.

Chapter 3
Requirements Elicitation

This chapter presents the requirements of the VisitCollage as elaborated during the initial conception of the system. First, it introduces the elicitation process that aimed to identify those requirements that would ensure proper behavior and maximize acceptability and then we report the findings categorized as functional and non-functional requirements.

3.1 Elicitation Process

In order to fully explore the problem, multiple brainstorming [30] sessions were conducted. During the brainstorming sessions, initial design concepts were produced and assessed by a domain expert and four HCI experts. In particular, the system's characteristics were discussed and analyzed in order to investigate to the maximum extend what functionalities one would expect from such a system and identify problematic areas that would require further elaboration.

Moreover, an extensive case study on technologically augmented exhibits was performed in order to identify the activity of visitors that needed to be recorded. In specific, the interactive systems of the exhibition "Macedonia from fragments to pixels" [31] located in the Archaeological Museum of Thessaloniki since 2010 were examined thoroughly. The exhibition comprises a series of interactive systems, by which the visitors have the opportunity to explore digital reproductions of ancient masterpieces. The digital content of these systems includes objects from the Museum's permanent collection and from Macedonia in general. Several of the artifacts presented are not available to the public, because of either their fragile state or their location.

The feedback gained from the brainstorming sessions in combination with the systems described in Chapter 2 and the extensive case study described above

was highly beneficial since it not only facilitated concept formulation but also specified the main features that the envisioned system should offer.

3.2 Functional Requirements

Functional requirements define what a system is supposed to accomplish and include calculations, technical details, manipulation and processing of the data and other specific functionality describing the input, behavior and output of the system. Generally, functional requirements specify what a system is supposed to do and usually expressed in the form of *"system shall do <requirement>".*

The functional requirements that derived from the elicitation process include the following:

- Registration facility: provide new users a means to subscribe to the system.
- Recognition through a unique id: upon registration, each user receives a unique code through which the system can discriminate the actions performed by different users.
- Multi-user support: the system should support the simultaneous interaction of more than one user.
- Multilingual content provision: during tours, the generated content should be available in the visitor's preferred language.
- Multimedia support: to enhance the users' experience, content should be enriched with information that can be easily perceived (e.g., sound, video, animations, etc.).
- Events disambiguation: for each incoming event, the system should discriminate in which action it corresponds.
- Platform independent visualization: the generated journals could be able for view from any platform.

- Remote access: each user should be capable of viewing their profile and their past visits remotely.
- Extraction of users' interests: the system should be capable of detecting and extracting users' interests from the systems visited.
- Provision of suggestions: the system should discover related content to the overall tour of the visitors and provide suggestions.

3.3 Non-Functional Requirements

In contrast to the functional requirements, non-functional requirements (*also known as quality requirements*) impose constraints on the design or implementation of the system. In broader terms, non-functional requirements specify how a system is supposed to be and usually expressed in the form of *"system shall be <requirement>".*

Non-functional requirements are divided into three categories: a) performance requirements, b) interface requirements and c) AmI requirements.

Performance requirements include:

- Ease of interaction: the interaction should be intuitive, easy to memorize and precise.
- The system should be self-explanatory: it should be easy to use for the first time after five minutes of training.
- Real time logging: each user action should be stored immediately by the system for further processing at the end of the visit.

Interface requirements include:

- Regarding the hardware: the browsing of the visit history requires at least a mouse and a keyboard for desktop users while for mobile users touch screen is required.

- As for the software: a modern browser is required to view the content and content is retrieved from sources that describe semantic information.

AmI requirements include:

- Unobtrusive hardware: use of smart devices, sensors etc. to capture users' activity
- Seamless communication and computer infrastructure
- Human-centric interfaces

Chapter 4
VisitCollage Overview

In the following sections, an overview of the VisitCollage system is provided. Initially, the data-model and the respective retrieval mechanisms are presented. Next, more details are reported regarding the communication protocol used for data exchange, the approaches adopted to determine user interests and the recommendation system. Finally, a short summary of the different visualization techniques that aim to increase visitors' engagement and maintain the "after-visit-experience" is given.

4.1 Data Representation

VisitCollage supports activity logging of the visitors within the exhibition space, interest detection during visit and delivery of personalized recommendations. For that to be achieved, the selected approach should offer: (i) rich semantic representation facilities, (ii) entity hierarchies, (iii) inference support, and (iv) "intelligent" information retrieval mechanisms. Subsequently, RDF [32] and RDFS [33] were selected for the information representation, SPARQL [34] queries for the information retrieval and the Euler engine [35] in combination with Notation3 [36] rules for the inference process.

4.1.1 Semantic Based Modeling

To overcome the extensive heterogeneity of the data sources within an exhibition space, a semantic data model was implemented. Semantic data models explicitly define relations between the data elements in the form of binary relations. Such relations are expressed as triplets : *<Subject, Relation, Object>*.

The benefits of building a semantic database are highly significant, as cross-domain integration can be achieved. This implies that VisitCollage holds no limitations regarding the context of use, as the systems' knowledge base can not

only be shared but also instantly aggregate content from well-established domains of knowledge.

4.1.2 RDF

The VisitCollage system builds its knowledge base using the Resource Description Framework (**RDF**), which is a family of World Wide Web Consortium (**W3C**) specifications originally designed as a metadata data model. It evolved as a general method for conceptual description or modeling of information that is implemented in web resources. The RDF data model is similar to classic conceptual modeling approaches, such as **Entity-Relationship** or **Class diagrams**, as it is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, known as triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object.

4.1.3 RDF Schema

RDF Schema (**RDFS**) is a set of classes with certain properties and using the RDF language provides the basic elements to describe RDF vocabularies (otherwise called ontologies) and structure RDF resources.

RDFS constructs are RDFS classes and they are much like classes in object oriented programming languages. This allows resources to be defined as instances of classes, and subclasses of classes. Classes are themselves resources and are usually identified by URIs. A resource may be stated to be a member of a class using the *"rdf:type"* property.

Properties in RDFS are relations between subjects and objects in RDF triples, otherwise called predicates. All properties may have a defined domain and range. The domain of a property declares the class of the subject in a triple whose second component is the predicate. The range of a property declares the class or datatype of the object in a triple. The taxonomy of classes is formed by the property

"rdfs:subClassOf" and the taxonomy of properties is formed by property *"rdfs:subPropertyOf".*

4.1.4 Protégé

To develop the hierarchies needed for the entities, Protégé [37], an open source ontology editor and knowledge base framework is used. Protégé offers a suitable environment for ontology definition, data insertion and formation of SPARQL queries. VisitCollage entities are represented inside Protégé in classes containing suitable properties to fit the needs of information storage and retrieval. In Figure 1, an overview of the main classes and subclasses hierarchy of VisitCollage ontology is presented Figure 1 as visualized by the Protégé module named Jambalaya [38].

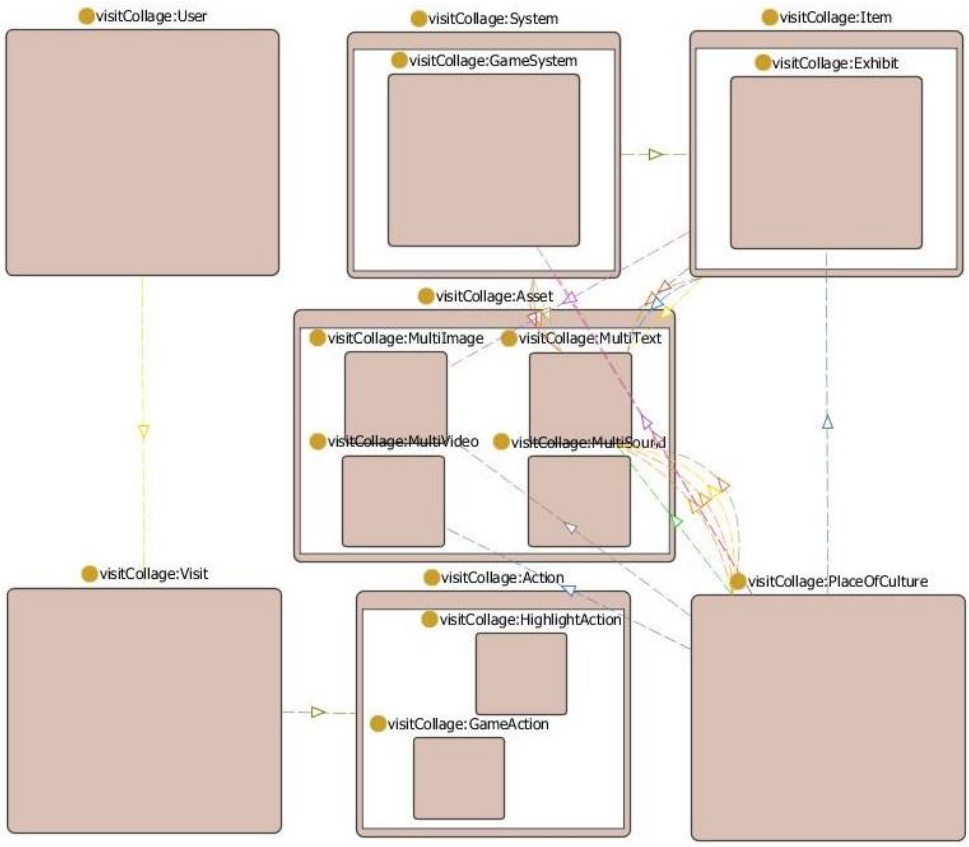


Figure 1: Ontology main classes' hierarchy

4.2 Querying and Reasoning

4.2.1 SPARQL

Data stored in the VisitCollage ontology are retrieved through semantic queries (i.e. an RDF query language). The SPARQL query language provides a wide support of statements including conjunctions and disjunctions, required and optional values in order to get results that may satisfy an obligatory or optional condition and support for constraining queries to limit results within specified thresholds.

SPARQL is built on the triple pattern; as a result, SPARQL queries contain a set of triple patterns. Triple patterns are like RDF-triples, except that each of the subject, predicate, object may be a variable. Each pattern matches a sub graph of the RDF data when RDF terms from the sub graph may be substituted by the variables and the result is returned in an XML document format.

Prefix	`PREFIX visitCollage: <http://www.ics.forth.gr/ami/culture/visitCollage#>`

Query	`SELECT ?firstname`
	`WHERE {`
	`?user rdf:type visitCollage:User .`
	`?user visitCollage:userFirstName ?firstname .`
	`?user visitCollage:userId "user0012"^^xsd:string .`
	`}`

Table 1: Example SPARQL query

Table 1 presents a simple SPARQL query that aims to find the first name of a user with a specific id within a given data set. The query is separated into two parts: the prefix part where the preferred abbreviation to the base URI of our ontology is defined and the query part where the *SELECT* clause identifies the variables to appear in the query results and the *WHERE* clause provides the triples' pattern to match against the data graph.

4.2.2 Semweb.NET

SemWeb.NET [39] is a Semantic Web/RDF library written in C# for Mono or Microsoft's .NET. The library is capable of reading and writing RDF in both RDF-XML and Notation3 format. SemWeb.NET provides mechanisms for keeping RDF in persistent storage (memory, MySQL, etc.), querying via simple graph matching and support for SPARQL queries, on both local and remote endpoints. Moreover, SemWeb.NET provides RDFS and rule-based reasoning over the data that are stored in the data-model, based on the Euler proof mechanism [40].

Euler mechanism is an inference engine supporting logic-based proofs. It is a backward-chaining reasoner (working backward from the goals) enhanced with Euler path detection. The Euler engine is used to generate inferences based on the facts that exist in the data-model, the taxonomies that have been defined for the entities and finally, a set of rules with Notation3 description framework. The implemented rules exist in external files separating the decision logic from the program itself thus simplifying future extensions or modifications.

Inference mainly takes place to the purpose of extracting visitors' interest inside the exhibition space. Another feature of the SemWeb.NET library used in VisitCollage includes RDF/XML reading and writing in order to: (i) populate the data-model with data, (ii) delete temporary data and (iii) modify at real-time existing data. Figure 2 shows an example of a rule for the Euler engine expressed in the Notation3 format. This example rule fires when a visitor has seen an image for more than five seconds and as a result increases the visitor's interest weight about that image.

```
{
  ?visit          a                                    visitCollage:Visit.
  ?visit          visitCollage:visitId                 "visit0102"^^xsd:string.
  ?visit          visitCollage:visitVisitedThings      ?visitedthing.
  ?visitedthing   visitCollage:visitedthingTypeOfAction "OpenClose"^^xsd:string.
  ?visitedthing   visitCollage:visitedthingAssetType   "Image"^^xsd:string.
  ?visitedthing   visitCollage:visitedthingMaxTime     ?maxtime.
  ?maxtime        math:greaterThan                     "5"^^xsd:int.
}
=>
{
  ?visitedthing   visitCollage:rule1ACommand           "increaseInterestWeight"^^xsd:string.
}.
```

Figure 2: Example rule using Notation3 syntax

4.3 Systems Interconnection

To fulfill its intercommunication needs with the connected systems, VisitCollage relies on a generic services interoperability platform, namely FAMINE (i.e. FORTH's AMI Network Environment), which has been implemented in the context of the ICS-FORTH AmI Programme. Famine provides the necessary functionality for the intercommunication and interoperability of heterogeneous services hosted in an AmI environment. As stated in [41], FAMINE meets the requirements that a middleware system for Ambient Intelligence should address. These requirements involve heterogeneity integration, synchronous and asynchronous communication, resilience, security and ease of use. FAMINE is based on CORBA and encapsulates mechanisms for service discovery, event driven communication and remote procedure calls, supporting a wide number of programming languages and frameworks, such as Java, .NET languages family, C++, Action Script etc. The services running using the middleware may be spread across the network and programmers do not interfere with network connection establishment. On the contrary, developers focus only on the functionality that each service should offer, implementing the desired interfaces as a service and creating wrapper classes that can handle incoming data for any client that receives it. Figure 3 shows an overview of the FAMINE's architecture.

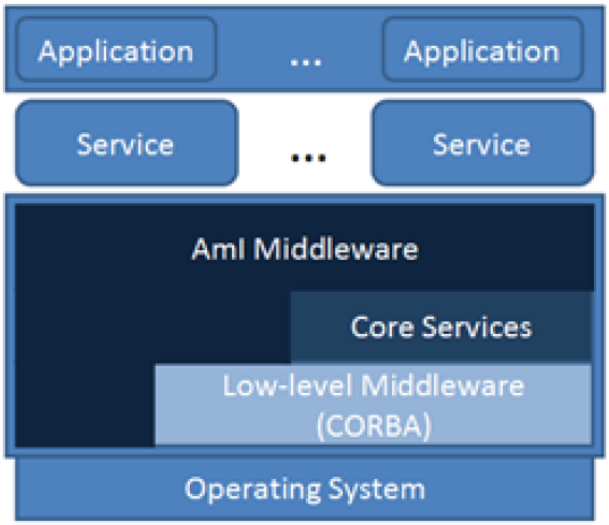


Figure 3: Architecture of FORTH's AMI Network Environment (FAMINE)

4.4 Activity Monitoring

VisitCollage relies on the aforementioned intercommunication mechanism to monitor visitors' activity within the exhibition space and collect the necessary data that will eventually facilitate the generation of the visit collage. For that to be achieved, the actions that need monitoring and their translation to appropriate FAMINE events should be determined.

In order to identify which actions needed to be recorded, an extensive research on technologically augmented exhibits was performed to reveal the available options. In more details, the interactive systems of the exhibition "Macedonia from fragments to pixels" [31] were examined thoroughly.

The key finding of this process was that the content of each system could be systematically categorized into "Items" (thematic areas presented by the system) and "Assets" (the multimedia that each thematic area includes). This categorization was essential not only for the representation of the heterogeneous systems' data into VisitCollage but also for the determination of actions that could be performed by the visitors. Taking all the above into consideration, the interactions that take place on those systems were classified in four main categories:

- **Open / Close:** This set of actions indicates the activity when a visitor inspects or moves away from a specific *Item* or *Asset.*
- **Maximize / Normal:** This set of actions indicates the activity when a visitor maximizes an image or brings it back to normal.
- **HighlightOn / HighlightOff:** This set of actions indicates the activity when a visitor starts or stops highlighting a specific part of an *Asset* e.g. a specific area of an image or a specific time period of a video.
- **Game:** This type of action indicates that the visitor has finished playing a game incorporated in a *GameSystem.*

As far as the transmission of actions is concerned, VisitCollage incorporates a specific component called *Actions Logger* in order to handle the actions send from the connected systems in the form of events through FAMINE. The *Actions Logger* component observes the connected systems and stores any user-driven interactions in the ontology. Figure 4 presents how the interoperability between the smart exhibits and VisitCollage is accomplished.

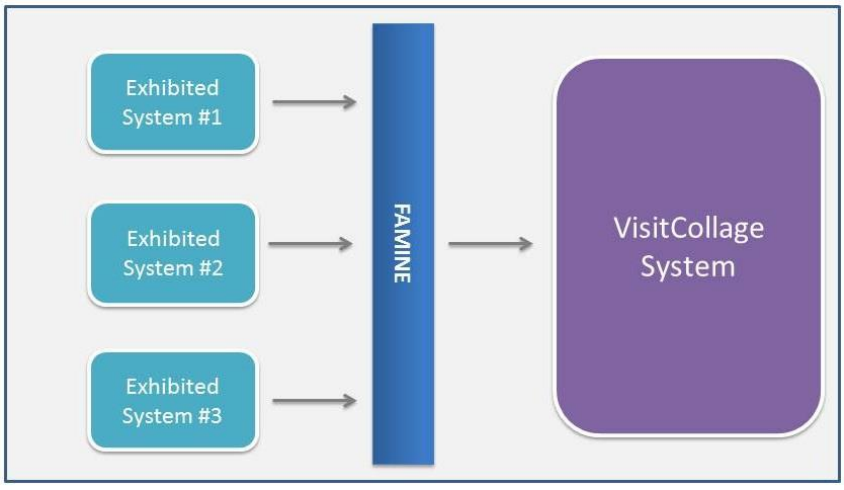


Figure 4: Smart exhibits and VisitCollage interconnection

4.5 Extracting Visitors' Interests

Upon visit completion, after the performed actions have been transmitted to VisitCollage, the process of discovering the visitor's interests begins. As the visitor's actions have been stored in the data-model in the form of RDF-triples, the *Interests Reasoner* component of VisitCollage is responsible for retrieving and processing the stored actions for the specific visit.

The *Interests Reasoner* initially processes the complete actions list in order to merge the complementary actions (e.g., open-close pairs on the same asset) and collect statistics about the *Items* and *Assets* visited. These statistics include, among others, the number of times one visited an *Asset* or *Item*, the total time spent, aggregation of the maximum time spent (if visited more than once) etc. All data gathered from the preprocessing are initially stored in internal data structures for immediate access and as soon as preprocessing is over they are stored back in the persistent data-model.

Thereafter, a set of predefined rules is evaluated to infer the interests of the visitor based on the latest interaction history. The rules are implemented and stored in a separate external file to decouple the decision logic from the actual system and facilitate maintenance and scalability. During the inference process, the reasoner increases the interest weights of an a*sset,* an *item,* a system or place accordingly, and at the same time collects the URIs of the mostly-viewed *tags* included in the respective metadata. Tags are meta-information included in *items*, *assets* and *places* used to provide further descriptions about these in the form of URIs. Moreover, the reasoner collects general statistics for that visit such as total number of images viewed, total number of videos considered interesting, etc.

Finally, all the dynamic data collected at runtime through interaction monitoring is encoded into behavioral models that facilitate the adaptation of the filtering process. These models along with the data convey further information about the visited artifacts and stored by the *Interests Reasoner* in the data-model for further analysis.

4.6 Exporting Recommendations

One key feature of the VisitCollage system is the delivery of personalized recommendations based on the visitors' assumed interest in each individual *Item* explored. The *Content Manager* is the component that exploits the information generated by the *Interests Reasoner.* The *Content Manager* follows a two-step procedure: first, it classifies the *Items* visited based on their interest weight calculated during the reasoning phase, and then generates personalized suggestions for every "interesting" *Item* along with other related places worth visiting.

During the first phase, the *Content Manager* retrieves the stored data, related to the visited items from the ontology and orders them in descending order by their interest weight calculated during the reasoning phase. The collected *Tags* are also retrieved and ordered in descending order based on their appearances count.

The generation of recommendations begins after the classification phase finished. During this phase, the *Content Manager* iterates over the ordered *Items* list and searches for any related *Items* that the visitor has not visited during the tour. Related physical exhibits to the ordered *Items* are also examined. The extraction of recommendations is a recursive process, where the *Tags* discovered from the visitor's tour among the smart exhibits are the key metadata based on which the recommendations are extracted. This process is repeated until the desired number of related *Items* and physical exhibits is reached. Upon completion, the *Content Manager* based on the discovered *Tags* searches for related places that meet the user's interests. The number of recommended artifacts in both cases is variable and is configured separately from the component to support scalability.

As a final step, the *Content Manager* dynamically generates the appropriate data structures in the data-model and persistently stores every interesting *Item*, its respective *Assets* (images, videos, texts, sounds) and the inferred suggestions for further use during visualization.

4.7 Visualization

One of the major requirements for VisitCollage was the need for platform-independent visualizations accessible from anywhere. For that to be achieved, a web-based approach was selected as such kind of cross-platform compatibility can be supported through a web browser that every desktop or mobile device embeds nowadays.

HTML (HyperText Markup Language) [42] is the main markup language and the one used for creating web pages and other information that is displayed in a web browser. Additionally, in order to enrich the interaction the JavaScript [43] programming language was also used for specific features such as maximize images, view the image gallery etc. Moreover, CSS (Cascading Style Sheets) [44] is the style sheet language used for describing the look and formatting of the html document and by this way the separation of the document content from the presentation is achieved.

The *Presenter*, as implied by its name, is the VisitCollage system component, which undertakes the process to visualize the users' experience in the exhibition space. The captured content is visualized in three different forms: (i) sequentially, where the content is ordered either chronologically or by interest, (ii) spatially, where the interactive systems visited are overlaid on top of a floor plan of the exhibition and (iii) as an elegant booklet where the content is arranged within its pages. The *Presenter* incorporates separate files, called templates, for each form of presentation and for each different component of the interface. In particular, it includes HTML files for the design of the header of the web page, the footer, the place visited, the related content and one for each *Item* to be presented depending on the type of multimedia (image, video, text, sound) the visitor liked the most. Every HTML template includes special placeholders that are populated to deliver dynamic content.

Initially, the *Presenter* retrieves the data stored by the *Content Manager* from the data-model using properly formatted SPARQL queries and loads it into its

internal data structures. Afterwards, the *Presenter* iterates over the retrieved data set, selects the appropriate template, populates the placeholders with actual data and subsequently generates the final HTML file. The visitor can access the generated web page at any time and from anywhere. Figure 5 presents the overall process followed by VisitCollage in order to deliver the final content based on the visitors' actions.

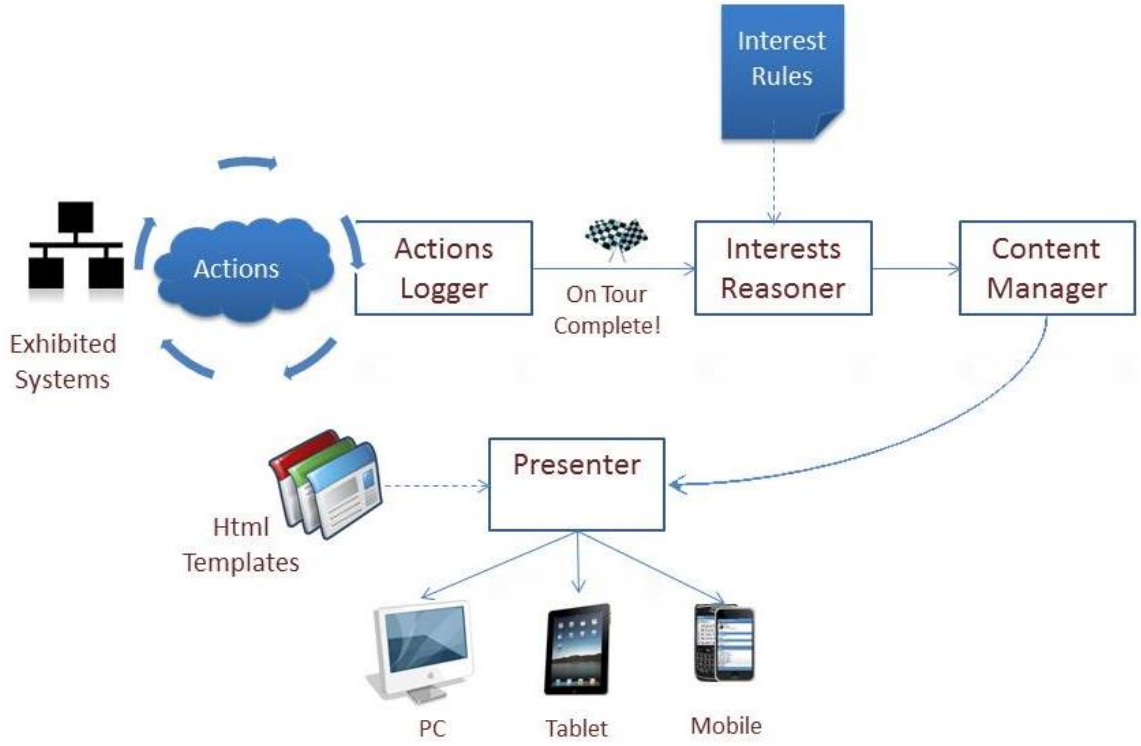


Figure 5: VisitCollage workflow

Chapter 5
System implementation

5.1 Architecture

The VisitCollage system proposed on this thesis aims to enrich the visitors' experience in an ambient exhibition space in a transparent manner. It provides all the necessary mechanisms for each different process that needs to be executed in every phase, so as to constitute a complete system that every ambient exhibition space could integrate. These include activity monitoring, interests' detection, personalized recommendations and visualization. Therefore, the offered services take into consideration both the needs of the users and the need for establishing well-defined structures and mechanisms for the data to be hosted, extracted and manipulated.

In particular, the system provides the functionality for users to subscribe into the system and the possibility to dynamically populate the data-model with information about new exhibition spaces and systems. Moreover, VisitCollage facilitates the recording of the users' interaction with the exhibits in order to capture their interests and behavior by offering transparent ways of intercommunication between the systems, in such a way that the visitor's tour among the exhibits is not affected in the slightest. In addition, VisitCollage incorporates methods for extracting the content the visitors found interesting during their tour, as well as for exportation of personalized recommendations suitably adjusted to the visitor's interest. Finally, VisitCollage encapsulates methods to organize and present the interesting content as obtained during the visit along with recommendations both perfectly suited in the form of web pages in order for those to be accessible even from virtually every location (e.g. home, office, mobile, school etc.). In this way, VisitCollage aims to deepen visitors' experience, increase their satisfaction, and provide them motivation for multiple visits.

VisitCollage is built upon a modular architecture. The VisitCollage's core consists of six major components: the *Systems Manager*, the *Users Manager*, the *Actions Logger,* the *Interests Reasoner,* the *Content Manager* and the *Presenter* orchestrated by the *Main Coordinator.* Each one of the six major components is deployed as a single instance class following the Singleton design pattern. The Singleton pattern [45] ensures that a class has only one instance and that the instance is easily accessible. In addition, the Abstract Factory design pattern was applied in order to provide greater flexibility regarding the instantiation of the above classes. In particular, we have defined an interface for each component exposing its necessary operations and a factory class, which realizes the instantiation of the concrete class that implements the interface. Subsequently, the *Main Coordinator*, which is the component that instantiates the above components does not hard-code the desired instances; instead, it invokes the create method for each component and remains unaware of which concrete class is instantiated. Additionally, there are a number of utility modules offering auxiliary services and functionality to the main components of the system such as the *Ontology Manager*, the *Triples Converter* and the *Event Handlers* modules. Figure 6: VisitCollage architecture shows the architecture of VisitCollage. In the following sections, an extended description of each component and their interplay will be provided.

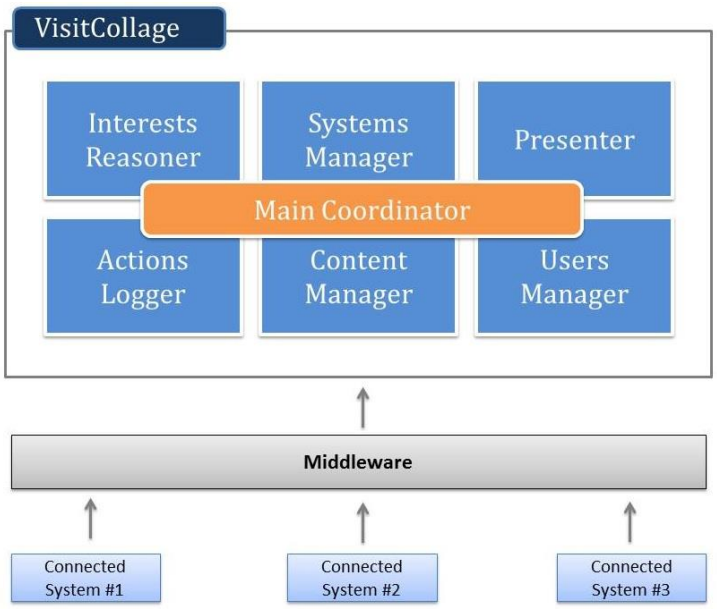


Figure 6: VisitCollage architecture

5.2 Main Coordinator

An ambient exhibition space is an intelligent environment consisting of "smart" systems that VisitCollage aims to exploit in order to enrich the visitors' experience. The *Main Coordinator* is the component that orchestrates the whole process and constitutes the entry point through which the connected systems can exchange data with. This component initially activates all the components of VisitCollage and is aware of their status. The *Main Coordinator* also provides the handlers for listening to the events transmitted to VisitCollage through the *Event Handlers* module it integrates.

5.2.1 Events Handling

The *Event Handlers* module of *Main Coordinator* component is used to listen and handle the incoming events from the connected systems. These events either come from the connected digital exhibits regarding the activity of the visitor, or from other devices (e.g. computers at the entrance of the exhibition, sensors) in order to subscribe a new user, populate the data-model with data about a new exhibition space or indicate that a user has started or ended a visit.

Once the *Event Handlers* module receives a message with all the information about a new user, it initially stores the profile image of the user (if provided) at the location specified in the configuration file of the VisitCollage. Afterwards, it creates a new *User* instance populating it with all the information the user has provided such as first name, last name, age, profile image filename etc. and calls the *Users Manager* to store the *User* data structure in the data-model.

Regarding the events for subscribing a new exhibition space, the data received are encoded in an XML format. The handler for such messages temporarily saves the data in an XML file and notifies the *Systems Manager* to populate the data-model with those data providing the path to the XML file.

As far as the interaction with the smart exhibits is concerned, a unique identification is required in order to recognize each user's activity. This unique id can originate from an ID card, RFID card, etc. The system holds no limits about the origin of the identification method, thus providing a flexible manner by which the visitors can be identified in an exhibition space.

The *Event Handlers* module also provides handlers for events when users start a new visit at an entry point or terminate their visit respectively. The messages received in both cases incorporate the unique ID assigned. In case of a new visit, the handler notifies the *Actions Logger* providing the unique ID from which the event arrived in order to create a new *Visit* instance for this visitor via the *Users Manager* and update its dictionary structure that holds the currently logged-in visitors. During a tour, every step is recorded through an event that associates a single action with the ID of the visitor who performed it. Then, the corresponding handler populates the appropriately created data structures for each action type and notifies the *Actions Logger* to store that action for that specific ID. Finally, upon visit completion, the *Actions Logger* is notified and subsequently the *Interests Reasoner,* the *Content Manager* and the *Presenter* components are invoked sequentially to augment and imprint the enriched experience.

5.3 Systems Manager

The *Systems Manager* is the component responsible for data manipulation as it offers the appropriate mechanisms both for storing and retrieving data from the data-model supported by the SemWeb library. Specifically, the *Systems Manager* enables the insertion of data into the system that describe attributes of the exhibition space, the systems exhibited and all the information that these systems incorporate. Moreover, this component also offers data manipulation-related functionality, thus every component communicates with the *Systems Manager* in order to access the exhibit-related data. Towards this end, the *Systems Manager* integrates two assistive modules, the *Ontology Manager* and the *Triples Converter* module.

5.3.1 Ontology Manager

The *Ontology Manager* module provides all the necessary functionality to handle the data of the system. The *Ontology Manager* is deployed as a single instance class following the Singleton design pattern in order to provide ease of access to its functions from the other components across the entire system. This module offers the ability to load the ontology file in memory for fast access via the "import" method of SemWeb library. It also provides methods to store the in-memory model back to the ontology in order to achieve permanent storage.

Built-in functions offered by the SemWeb library facilitate data manipulation (i.e., Add, Remove methods) in the form of triples. In addition, the data within the data-model maintained by the *Ontology Manager* can be queried. This is possible not only through the use of SPARQL but also through the RDFS reasoning supported by SemWeb (Select, Contains methods) over the data-model. The add and remove functions are mainly used during the registration of users or systems, the logging of visitors' activity and even in the reasoning phase to infer the visitors interests where data are added or removed based on the rules evaluation. The queries are mainly used during the extraction of related content and visualization in order to make the data available for presentation.

5.3.2 Triples Converter

One of the features of VisitCollage is the dynamic integration of new data about an exhibition space. These data include all the information about the place, the systems and the physical exhibits included within this place as well as all the information about the multimedia the above elements integrate. All the above data are organized within an appropriately structured XML file.

Once an event for storing a new set of data about a place is received, the *Main Coordinator* notifies the *Systems Manager* to store the data by forwarding the path to the XML file included in the newly received *store* event. Since the file is accessed, the *Systems Manager* deserializes the data into suitably formed classes. Figure 7

shows the class diagram of the classes involved to store the data of an exhibition space. The *Triples Converter* undertakes the process of installing the new data to the data-model. To do so, it converts the data into triples and in cooperation with the *Ontology Manager* imports those triples into the data-model.

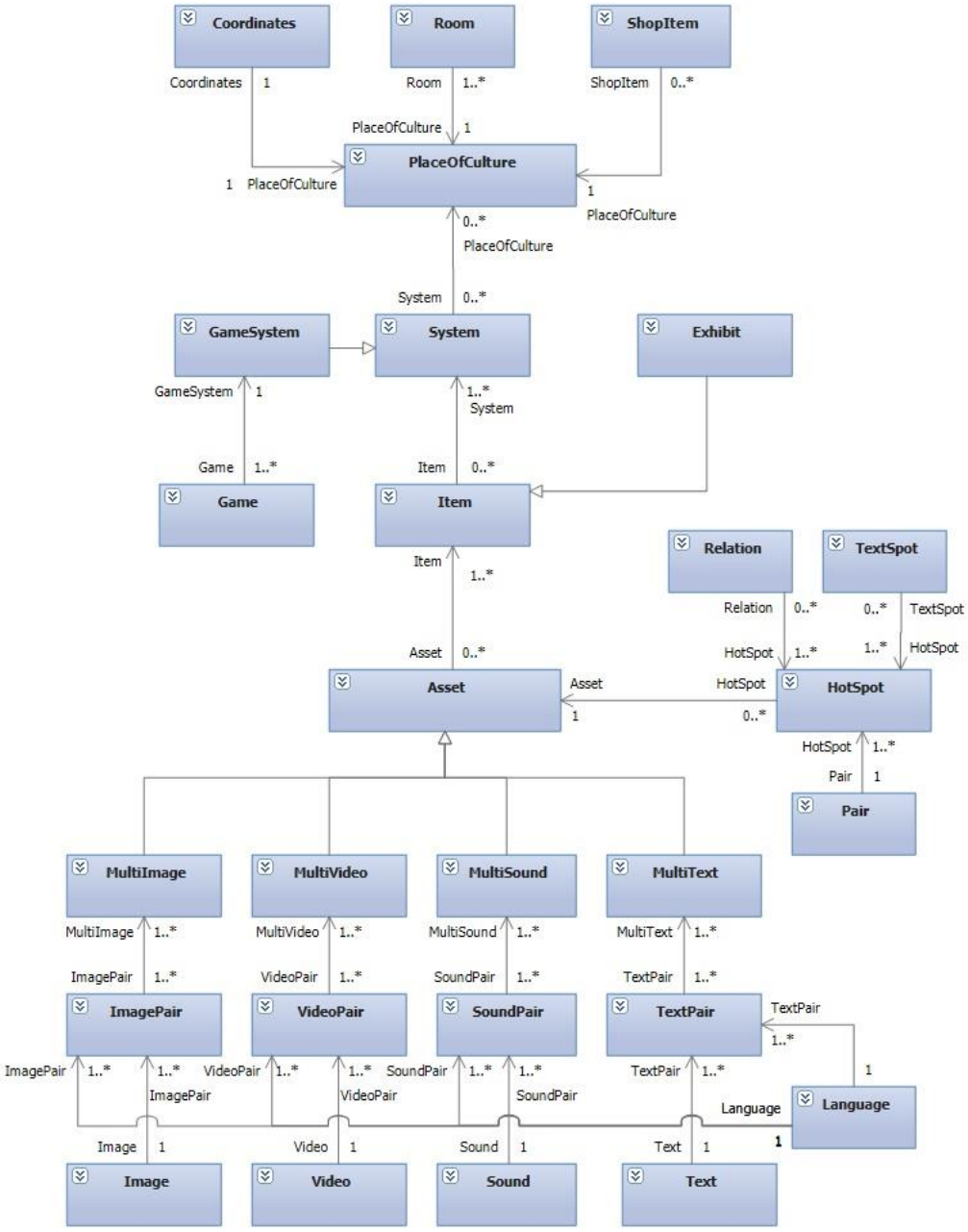


Figure 7: Exhibition space class diagram

The *Triples Converter* algorithm is a recursive process on the data classes. The algorithm loops over the class instances and using the .NET reflection mechanism

"discovers" their primitive data, the objects and the data structures and forms suitable RDF triples, which are then populated into the data-model by the *Ontology Manager.* Every class instance is either the subject or the object of an RDF triple, while every property corresponds to the predicate; however, when an RDF property starts from a class and points a literal value, then the object of that triple is an RDF Literal instance.

Figure 8: Populating the data-model with exhibition space data

5.3.3 Taxonomies

A taxonomy is a particular classification of things or concepts, as well as the principles that underlie the classification arranged in a hierarchical structure. Taxonomies are a fundamental part of the Semantic web, as they enable intelligent agents to make logical inferences and retrieve information. As semantic web notion prompts for more structured and connected data, two external popular vocabularies are used in the VisitCollage ontology, the FOAF [46] ontology and the CIDOC Conceptual Reference Model [47]. Using semantic web vocabularies permits intelligent agents to make sense of the entities appearing in ontologies and the connections between them.

FOAF (Friend Of A Friend) ontology describes persons, their activities and their relations to other people and objects. In the VisitCollage ontology, some of the elements of FOAF such as first name, last name and gender are used for the *User* entity. The CIDOC CRM [48] provides definitions and a formal structure for describing the implicit and explicit concepts and relationships used in cultural heritage documentation. In the VisitCollage ontology, resources for the description of characteristics of physical exhibits are used. These include an exhibit's current

location, the location where it was retrieved, the date found, the materials out of which it is composed by and the chronological period found.

The taxonomy is perfectly suited to the VisitCollage needs, allowing efficient classification and retrieval of the content. Since the taxonomy's structure is encapsulated in the contained concepts and their relationships, VisitCollage employs the RDFS technology to define taxonomies. A concept is modeled as an RDF class and a relation between two concepts is modeled as a property between the respective classes; the taxonomy's hierarchy is implicitly defined through classes' inheritance. The taxonomy's data are represented as RDF instances encoded in an RDF file along with the taxonomy schema. The taxonomy's schema was created using Protégé and the main classes that the Systems Manager populates with data are presented in the following sections.

5.3.3.1 PlaceOfCulture

The *PlaceOfCulture* class models the concepts that each ambient exhibition space integrates. The main properties of this class include:

- **Id**: a unique id used to identify each instance of the class
- **Name**: the name of the exhibition
- **Description**(short and long) : a brief and an extensive description of the exhibition
- **Type**: the category of the exhibition depending on the concept of the exhibited items (e.g. Architectural, Science etc.)
- **Website**: the URL of the exhibition's website
- **Photos**: instances of the *Image* class regarding photos of the exhibition
- **Rooms**: instances of the *Room* class including information about the exhibition's rooms (e.g. name, number, floor etc.)
- **ShopItems**: instances of the *ShopItem* class, which provides information about the items of the potential shop of the exhibition (e.g. title, description, price, quantity etc.)

Regarding the location of an exhibition space, the properties used include the address, the coordinates and the location of the exhibition, this last property is a resource pointing to the CIDOC CRM. Moreover, an ambient exhibition space includes systems and physical exhibits as well. For this reason, the *PlaceOfCulture* class incorporates properties that refer to the *System* and *Exhibit* classes described below.

Finally, in order to extend the description of a place two additional properties are introduced; namely tags and relations. The tags are URIs pointing to concepts from external or internal resources providing further knowledge to the place modeled. On the other hand, the relations property points to instances of the *Relation* class. This class consists of three properties: the subject, the property and the object, which are also URIs and the aim is to provide a way to describe relations like RDF triples between the tags. In addition to extending the knowledge of the class, tags and relations properties are the key properties for the discovery of related content to what the user liked most during their visit.

5.3.3.2 System

The *System* class incorporates the properties needed to model the smart exhibits within the exhibition space. These include a unique id for the system, the title of the system, a description and the place the system is located. In addition, the *System* class contains the property named items that points to the individual *Item* instances it includes.

The *GameSystem* class extends the *System* class in order to model a system that incorporates one or more games. For this reason, the only property used, along with the properties inherited by the *System* class, points to instances of the *Game* class that represents all the info about a game included in the system. The properties of the *Game* class are the title of the game, the description and all the multimedia assets the game consists of.

5.3.3.3 Item

The content of smart exhibits is organized into “Items” which represent the thematic areas presented by those systems. For instance, if a system presents a number of crowns that belong to the king “X”, the *Items* would be the crowns of king “X”. The concept of the *Item* is introduced not only for grouping the data presented by the different systems but also for promoting the notion of the “virtual exhibit” incorporated in the systems.

The VisitCollage ontology stores the “virtual exhibits” integrated into a system in the form of an instance of the *Item* class. Each *Item* contains properties related to its id, its key photo, its short and long description, the systems in which it is embedded, as well as the typical interaction time specifying the amount of time a visitor needs to discover most its content. Moreover, the class embodies a property pointing to the supplementary multimedia assets of that item, such as additional images, videos, texts and sounds. In addition, as in the *PlaceOfCulture* class the properties tags and relations are used to both enrich the knowledge for an *Item* and provide the means for related content mining.

To differentiate a physical from a virtual exhibit, the *Exhibit* class is used. It extends the *Item* class thus; it inherits all of its properties and additionally includes more properties that point to the CIDOC CRM [47] resources as exhibit’s current location, the location found, the date found, the materials it is composed of and the period when it was found. Finally, a last fundamental property of the *Exhibit* class is the room id in which the exhibit is currently located within the exhibition space.

5.3.3.4 Asset

The *Asset* class is an abstract class used to model all the multimedia content of the VisitCollage system. Its direct subclasses are: (i) the *MultiImage* class, (ii) the *MultiVideo* class, (iii) the *MultiSound* class and (iv) the *MultiText* class. Each of them integrates a property of Multimedia class-Language pair that points to the properly internationalized multimedia class (i.e., image, video, sound, text). The *Language* class resource properties refer to the characteristics of the language, such as the name and the abbreviation. The supported multimedia types are the *Image, Video,*

Sound and *Text* classes modeled appropriately to describe the respective multimedia content and are presented below.

The main properties of the *Image, Video* and *Sound* classes include the following:

- **Title**: the title of the respective multimedia type
- **Filename**: the actual filename of the multimedia file
- **Description**(short and long): a brief and an extensive description about the respective multimedia type

The *Video* and *Sound* classes also incorporate the duration property, which is used to represent the duration of a video or sound file respectively. The *Text* class is modeled to store information about the texts included in the system. The properties used by this class are the actual text and the total words of the text, which is assigned dynamically by the system during the reasoning phase.

Each multimedia class inherits the id property along with the hotspots property from the *Asset* class. The latter points to instances of the *HotSpot* class. This class aims to model interactive spots that each one of the multimedia types could integrate. For an image, the interactive spots could be areas on the image that the visitor may interact. For a sound or image, the interactive spot could be a specific part of the file with specified duration. As for the text, there may be specific parts of the text that could be interactive.

For all the above multimedia types, the *HotSpot* class contains suitable properties to offer the ability to model such interactive areas on multimedia data. The properties of the *HotSpot* class are the following:

- **Id**: a unique identifier for each *HotSpot* instance
- **Tags**: URIs to convey further information about the specific spot of the multimedia
- **Relations**: instances of the *Relation* class, which provides the relations of the Tags

- **Shape**: instances of the *Point* class used to specify a number of points that consist an interactive area for an image
- **TextSpots**: instances of the *TextSpot* class indicating interactive spots of a text
- **Time**: a pair of numbers indicating the beginning and the duration of an interactive spot for a video or sound

Particularly, regarding the image, the shape property is a set of points that form an interactive area on the image. As for the video and sound, the time property is defined. It refers to a pair of numbers defining the point in the file where the interactive area starts and the duration expressed in seconds. Regarding the text, a property that refers again to a pair of numbers is used. The first number of the pair designates the letter from which the interactive area begins while the second indicates the length expressed as a number of characters beyond the first character of the interactive area. The *HotSpot* class incorporates the properties tags and relations in order to convey further knowledge to each hotspot of the multimedia.

5.4 Users Manager

The *Users Manager*, as its name indicates, is responsible for handling in cooperation with the *Ontology Manager* module any user-related data. As soon as the *Main Coordinator* receives an event for a new user subscription, the corresponding handler creates a new *User* instance and notifies the *Users Manager* to store that information. The *Users Manager* in turn checks if the id assigned to the user is valid and delegates the *Ontology Manager* to import the new user into the data-model using the *Triples Converter* module. Figure 9 presents the components and the modules involved in the complete subscription process.

Figure 9: Populating the data-model with user's data

Once a user starts a visit, the *Main Coordinator* notifies the *Actions Logger* component and provides the unique user id that will facilitate tracking. Using this id, the *Actions Logger* creates a new V*isit* instance for the specified user id via the *Users Manager*. The *Visit* class models all the necessary properties regarding the user's tour along the smart exhibits. VisitCollage supports both registered users and unregistered users. In case of an unregistered user the recording of their activity is also possible only with the unique id they are assigned for example from the reception of the exhibition space. In this case, the *Users Manager* when notified creates an empty profile for that user with the specific id and subsequently assigns a new *Visit* instance to the user.

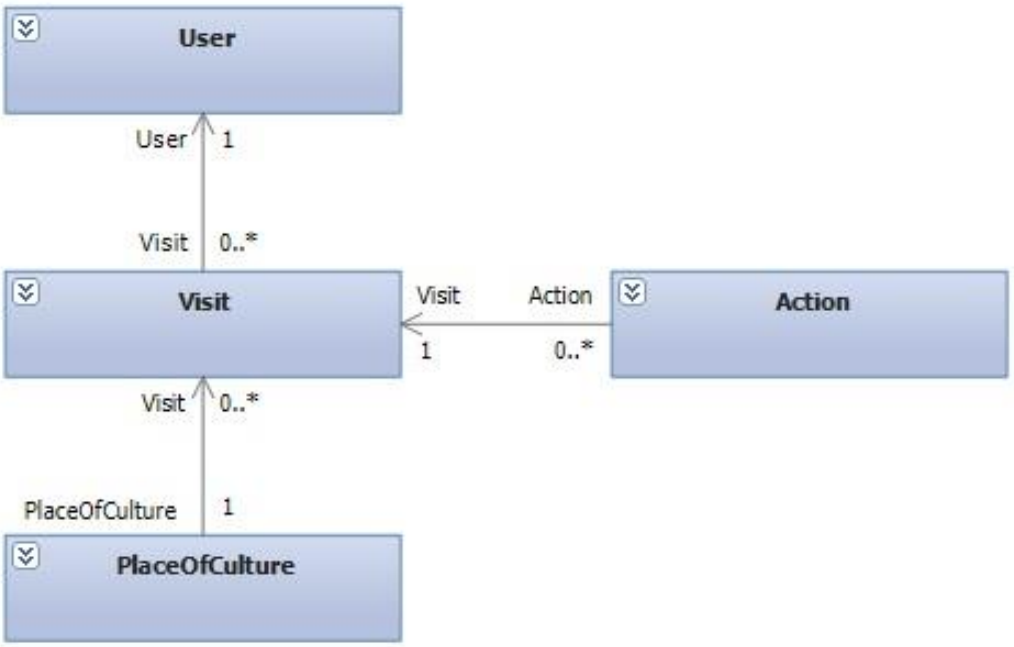


Figure 10: User and Visit classes

5.4.1 Users

The *User* class is used to model all the characteristics for a user of the VisitCollage system. In the implemented schema, the main properties needed to describe a person such as first name, last name and gender are coming from the FOAF ontology [46]. Additional properties include an id automatically assigned to the user, the date of birth, email, filename of a profile picture and the subscription date. Optionally, users are able to explicitly insert their personal interests via available options, which are then modeled as URIs in the *User* class similarly to the tags defined for *PlaceOfCulture, Item, Asset* classes. For example, if a user selects that he has an interest in science among other available options, this would match an appropriate URI that corresponds to the science concept and will be included in his profile.

The *User* class also incorporates the unique id that the users were assigned by which they are recognized during their visit among the smart exhibits. Furthermore, one more property included is the visits property. This property points to instances of the *Visit* class, so as to model every single visit that the user carries out along with all the information about these.

5.4.2 Visits

The *Visit* class incorporates all the properties needed in order to sufficiently store both the incoming information during the tour and the data exported once the visited is terminated. Properties stored in the *Visit* class include the following:

- **Id**: an automatically assigned id to each *Visit* instance
- **DateTime**: the date and time the visit took place
- **Duration**: the time the visit lasted expressed in seconds
- **Place**: The id of the visited exhibition
- **Impressions**: the impressions that the visitor may left in the form of a video

- **History**: the *Action* instances formed during the interaction

Moreover, the *Visit* class stores all the information that comes from the pre-processing of the actions such as the games played during the visit, what was opened, highlighted, maximized etc. Additionally, all the inferences extracted during the reasoning phase, such as how many multimedia of each type were viewed or how many were considered interesting along with the interesting *Items* and *Systems* are also stored in appropriate properties. Finally, the *Visit* class stores all the information about the related *Items, Exhibits* and *Places* as exported by the *Content Manager* in order to be available for the *Presenter* component which is responsible to visualize all the information stored into the *Visit* class.

5.5 Actions Logger

VisitCollage logs visitors' interaction with the connected systems while taking a tour in the exhibition space. For this to be achieved, VisitCollage incorporates the *Actions Logger* component. Conceptually, a visit in any given exhibition space can be decomposed in three discrete stages:

1. Beginning of the visit from an entry point
2. Interaction with the smart exhibits
3. Termination of the visit

In each of the above steps, the *Events Handling* module of the *Main Coordinator* component is the recipient of the events that arrive from FAMINE. Thereafter, the corresponding handler of the module notifies the *Actions Logger* by calling its respective method.

5.5.1 Starting a Visit

To begin with, regarding the first step, as soon as a visitor begins their visit, the *Events Handling* module receives a "StartSession" event along with the unique id of the user that started the visit. Subsequently, the *Actions Logger* receives the id

of the user and after verifying that the user has not already started a visit, proceeds by notifying the *Users Manager* to instantiate a new *Visit* for this user providing the id. Once the creation of the *Visit* instance is successful, the *Users Manager* associates that to the user through the visits property of the user and returns to the *Actions Logger* the id that assigned to the *Visit* instance.

The Actions Logger maintains a dictionary structure in order to recognize the currently active users in the exhibition. In this dictionary pairs of <users unique id, visit id> are maintained. Through this dictionary, the *Actions Logger* performed the verification mentioned above in order to ensure that the user is not already logged in. Since the *Actions Logger* has received the visit id created for a user that starts a new visit, it associates those values in a unique pair. In case a user terminates their visit, the respective pair is removed from the dictionary structure.

5.5.2 Interaction Logging

As soon as the user has successfully registered for a new visit, activity logging is accomplished via events dispatched from the connected systems whenever interaction occurs. The *Event Handlers* module of the Main Coordinator component handles the events that arrive through FAMINE. The events include all the information about the actions performed by the visitors and are encoded into appropriate classes from the respective handler of the *Event Handlers module.* Once the respective action class has been formed, the *Event Handlers* module notifies the *Actions Logger* to store the action providing alongside the unique id of the user that performed the action.

When the *Actions Logger* receives the action data, calls the *Ontology Manager*, which in turn transforms the data to RDF triples with the help of the *Triples Converter* in order to store the data into data-model. The *Ontology Manager* not only stores the action's data into the data-model but also updates the *Visit* class history property in order to log the action in the visit profile of the user. Figure 11 depicts the overall process of the users' interaction among the smart exhibits along

with the components involved. This recursive process terminates when the visitor finalizes their visit. The actions are further analyzed in the following section.

Figure 11: Interaction Logging Process

5.5.2.1 Actions

The VisitCollage system supports four discrete types of actions in order to record the visitors' activity. These include the following:

- Open / Close
- Maximize / Normal
- HighlightOn / HighlightOff
- Game

In order to cover the diversity of data regarding each type of action, a hierarchy of one superclass (i.e., *Action*) and two subclasses (i.e., *HighlightAction, GameAction*) have been created (Figure 12). The *Action* class incorporates the common

properties that an action should integrate, while the *HighlightAction* and *GameAction* classes inherit the basic properties from the *Action* class and include properties specific to the actions they model.

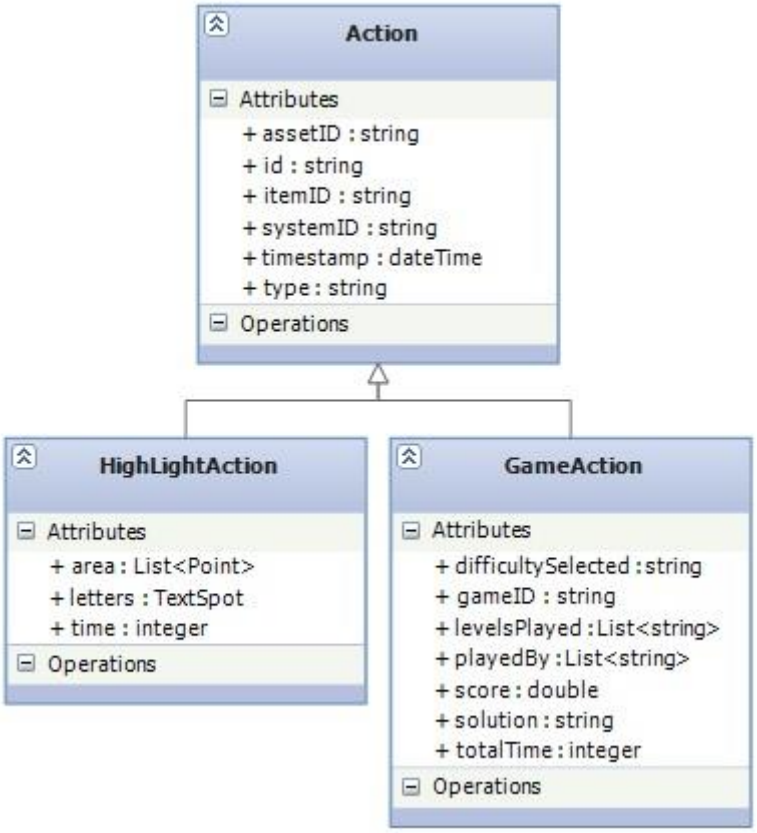


Figure 12: Actions Properties and Hierarchy

The conceptual mappings between user actions and the respective action types are presented below:

Open/Close: indicates that a visitor has just started or stopped viewing a specific Item or a particular Asset from an Item; thus an instance of the superclass Action is created with the type “Open” or “Close” respectively.

Maximize/Normal: indicates that a visitor has just opened a particular image at full screen or brought it back to normal size; thus an instance of the superclass is created with type “Maximize” or “Normal” respectively.

Other information that needs to be stored in the class is the *System's* id that the visitor interacts, the *Item's* id and the *Asset's* id when it comes to specific multimedia element. In case the action refers to Open/Close of an *Item* the asset id property is filled with a dash ("-").

In every case, the type property of the class is used to assign the respective string depending on the type of the action we instantiate. The timestamp and id properties are common across every action. Timestamp indicates the exact date and time the specific action occurred while the id property is automatically generated and assigned.

HighlightOn/HighlightOff: indicates that a visitor has started or stopped highlighting a specific part of an *Asset*; thus an instance of the *HighlightAction* class is created with type "HighlightOn" or "HighlightOff" respectively.

In particular, for images, a highlight action occurs when there is an area on the image that the user can interact with in order to see more details or view more information. Thus, the *HighlightAction* instance created in this case incorporates all the properties of the *Action* class and in addition, the area property of the class is used to indicate the part of the image that was highlighted. For videos and sounds, highlight means that the visitor played a specific part of the video or sound respectively. In this case, along with the inherited properties, the time property is incorporated in order to indicate the point in seconds the user started or stopped playing this part. For texts, the highlight action is applied in specific parts of the text (i.e., words or phrases) that the user interacts with in order to view more information. In this case, the *HighlightAction* class offers the letters property where a pair of integers indicating the starting and ending character of the text highlighted is stored. In each *HighlightAction* instance, one of those properties is evaluated depending on the *Asset* type it refers.

Game: indicates that a visitor has finished playing with a game; thus an instance of the *GameAction* class is created with type "Game".

Particularly, in case the user played a game in a *GameSystem*, VisitCollage receives a *GameAction* event. This event incorporates all the information from the interaction of the user with the game. These include the id of the game that the user played, the difficulty selected, the levels played, the score they achieved, the filename of the solution, how much time they played and the ids of the users that played in case of multiplayer game.

5.5.3 Terminating a Visit

Once a visitor terminates their visit, the *Events Handling* module of the Main Coordinator receives a "StopSession" event along with the unique id of the user that terminated the visit. Afterwards, the handler notifies the *Actions Logger*, which in turn removes the pair <user unique id, visit id> from the dictionary with the active sessions. Finally, it raises an event stating the *Main Coordinator* to begin the processing of the actions for the specific visit id that corresponds to the user that ended their visit in the exhibition space.

5.6 Interests Reasoner

A key feature of the VisitCollage system is the ability to generate inferences about the visitors' activity among the smart exhibits. Towards this end, a two-step procedure is followed by the *Interests Reasoner* component of VisitCollage as presented in Figure 13. The first step includes the preprocessing of the actions acquired during the logging phase. In this step, the system examines all the actions logged from each category and merges into appropriate class instances the pairs of complementary actions along with all the time-based information.

In respect to the second step, as soon as all the actions are processed and the respective classes formed, the inferences phase takes place. The logic of the inference process is defined by a set of rules written in the Notation3 format. The Euler engine exploits the rules and the data from the data-model in order to generate inferences about the visitors' interest. The rules formed in the context of

this thesis are indicative rules that highlight the potentials of the presented system, whereas an extensive evaluation is required to validate their accuracy and efficiency. In the following sections both the preprocessing phase and the rules used for the inference procedure are described.

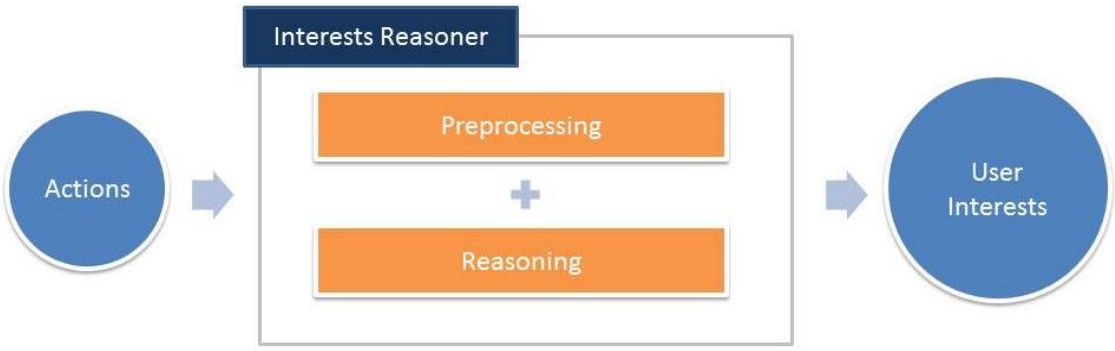


Figure 13: Processing Visitors' Activity

5.6.1 Actions Preprocessing

The preprocessing of actions is the process that follows the interaction logging of the users' activity among the smart exhibits. The *Main Coordinator* of VisitCollage when notified by the *Actions Logger* that the visitor has successfully terminated their visit instructs the *Interests Reasoner* component to extract the visitor's interests providing the visit id. The *Interests Reasoner* begins the process by merging the visitor's complementary actions into appropriate class instances in order to make them available for the reasoning phase. The actions merged during this phase include the Open/Close, Maximize/Normal, HighlightOn/HighlightOff type of actions.

Algorithm 1 is implemented in order to merge the complementary actions that the user performed and the logic is the following. First, the actions whose type property matches with the pair to be processed e.g., Open/Close are retrieved from the data-model with the use of a SPARQL query. Once retrieved, the actions are put into a List structure and ordered by the time occurred based on the respective property they incorporate. Next, each node of the List is examined starting from the second in order to find the closest that is complementary to the action of the

first node and their properties are equal. For example, if the first node includes an action of type Open and refers to system1, item1, asset1, it is expected to find the closest Close action with the same properties as the first one. Once two actions are found, they are combined into a new class instance with all the information about the pair and the nodes are removed from the List. In case that no complementary action is found, only the first node is removed. The above process is recursive and terminates when there are no elements in the List.

Function: ***MergeComplementaryActions***

Input: *ActionType1, ActionType2, VisitId*

Output: -

```
1: Retrieve ActionType1, ActionType2 from the data-model for VisitId
2: Store retrieved actions in actionsList[]
3: Order actionsList[] by time
4: WHILE actionsList[] has elements
5:   i = 0
6:   WHILE i<actionsList count
7:     IF (actionsList[i] type==ActionType1)and
8:        (actionsList[0] type==ActionType2)and
9:        (actionsList[i] same resources with actionsList[0]) THEN
10:       Merge(actionsList[0], actionsList[i])
11:       Remove actionsList[i]
12:       break;
13:    ELSE
14:       i=i+1
15:    ENDIF
16:  ENDWHILE
17:  Remove actionsList[0]
18: ENDWHILE
```

Algorithm 1: Merging Complementary Actions

The algorithm applies to all the complementary actions described above, except the Game action types, which are single actions with all the necessary information about the games the visitor played.

Regarding the merging of the Open/Close and Maximize/Normal type of actions, the *Interest Reasoner* creates instances of the *VisitedThing* class in order to store the information about each pair of actions. In particular, this class stores the id of the system that the actions came from, the id of the *Item*, the *Asset* id that the user performed the pair of actions and the date-time the activity occurred. All this information is available from the respective properties of the pair *Action* classes. Moreover, the *VisitedThing* class includes a type-of-action property where the type of the complementary actions is stored. This means that when grouping an action of type Open with a Close one, the "OpenClose" keyword is stored in the type-of-action property while for Maximize and Normal type of actions, the "MaximizeNormal" keyword is stored respectively.

Additionally, the class incorporates properties for time-based calculations as well as occurrences of the activity. Specifically, the total-time property is used for the time spent for this activity and initially is calculated as the time difference between the complementary actions. In case the same pair of actions is inspected on the same <system id, item id, asset id> triplet the time difference is added to the existing value of the total-time property. In parallel, the maximum value of the total-time value is calculated and stored in the max-time property in order to record the maximum duration of the interaction. Moreover, the *VisitedThing* class includes the times-occurred property, which records the number of times the specific activity occurred. Lastly, the interest-weight property is initialized to zero. The interest-weight property is used during the reasoning phase and increases accordingly based on the evaluation of the rules.

With respect to grouping HighlightOn/HighlightOff type of actions, instances of the *HighlightedThing* class are formed to store the information that comes from the *HighlightAction* classes. The *HighlightedThing* class extends the *VisitedThing*, thus it inherits all of its properties. Subsequently, those properties are handled in

the same way as described for the *VisitedThing* class except from the type-of-action property where the keyword "HighlightOnHighlightOff" is stored for this kind of interaction. In addition, three more properties are incorporated to store the required information regarding the *Asset* type the highlight action occurred. These properties are the area, letters and time that the respective *HighlightAction* instances incorporate.

Once the actions performed are merged into the respective *VisitedThing* and *HighlightedThing* class instances, the *Interests Reasoner* in cooperation with the *Ontology Manager* stores them into the data-model. These instances are pointed by the visited-things and highlighted-things properties of the *Visit* class created for the specific user.

5.6.2 Inferences Generation

In order to generate inferences about the users' interest, a set of predefined rules is evaluated. The rules are implemented in a separate external file to both separate the logic from the program and enable editing without the need for changing the program itself. The currently implemented rule set either calculates the interest shown by the visitor towards a certain asset/item/system/visit or extracts quantitative information (e.g., total images viewed) to assist interest estimation.

Regarding the interest shown by the visitor, the approach followed is to form appropriate rules that when evaluated will increase the respective interest-weight property. Specifically, regarding the visitors activity on the assets the rules mainly apply on the *VisitedThing* and *HighlightedThing* instances formed during the preprocessing phase. Particularly, the interest-weight property increases in the following cases:

- The max-time property of a *VisitedThing* instance with type OpenClose exceeds a threshold, which denotes that a visitor has viewed an asset for more than a specified period.

- The max-time property of a *VisitedThing* with type MaximizeNormal is over a threshold indicating that the visitor maximized an image for a sufficient period of time.

- The max-time property of a *VisitedThing* with type MaximizeNormal is greater than a threshold and has occurred more than once. This case indicates that the visitor not only viewed the image maximized but also repeated this action more than once.

- The times-occurred property of a *VisitedThing* instance with type OpenClose is greater than one and the average of total-time/times-occurred is above a threshold. This denotes that the visitor not only viewed the specific asset many times but also the average time spent was enough to cause their interest.

- The max-time property of a *HighlightedThing* instance for an image or text is greater than a threshold. This indicates that a visitor highlighted a specific part of the image or text for sufficient time to regard this part interesting.

- The time property of a *HighlightedThing* instance for a video or sound is greater than a threshold. This denotes that a visitor highlighted a specific part of video or sound, which means that they show interest in this particular part. Despite, in order to avoid to increment interest for very short parts we set a threshold to the length of such activity.

- The times-occurred property of a *HighlightedThing* instance is greater than one. This means that a visitor highlighted a specific part of an asset more than once.

- The visitor highlighted a specific area of an asset that lies inside a previous highlighted area of the same asset; thus the second has an added interest (i.e. consecutive highlights).

- In case of a *VisitedThing* instance with OpenClose type and the asset's tags match over a percent with the explicitly set interest-tags of a visitor. In other

words, if a visitor viewed an asset which tags match with their predefined interest-tags over a specified percent, most likely the visitor is interesting in this asset.

In order to estimate the visitors specific interests based on the assets or the hotspots they explored, a set of rules has been compiled to collect the tags of those assets that considered interesting. These rules are triggered whenever the interest-weight property of a *VisitedThing* or *HighlightedThing* instance becomes positive. Once evaluated, the tags of the asset or the part of the asset, in case of *HighlightedThing* instance, are stored in a dictionary structure with a key the URI of the tag and a value the occurrences of the tag. In case a tag already exists, the old value is incremented by one. In order to avoid common tags to gain high scores, the system eliminates them from the dictionary based on a list of common tags it maintains, similar to stop words [49]. This list is formed after the assets and items are inserted into the data-model for the first time. Tags with high frequency are annotated into this list, which in turn acts as a filter to the dictionary of the tags explored by the visitors.

Besides the rules created in order to assign interest on the assets visited, there are rules by which the system infers interest on the *Items* visited. In order to sufficiently model the information regarding a visited item, instances of the *ItemStatistics* class are created. The instances are instantiated when there is a *VisitedThing* instance that refers to an item. As previously stated when the *VisitedThing* asset-id property is equal to "-" (null/empty) this instance describes a visited item. Subsequently, the *ItemStatistics* class includes the system and item id properties along with the date-time that the item was visited populated with the values of the respective *VisitedThing* instance. Additionally, other information passed from the *VisitedThing* instance to the *ItemStatistics* instance include the total time spent, the times visited and the maximum time spent indicated by the max-time property. In addition, the *ItemStatistics* class incorporates the total-assets property, where the total number of assets for this item id is stored at the instantiation of the class. Moreover, the visited-assets and interesting-assets properties are included in order to store the number of assets visited for the

specific item id and the number of assets considered interesting during the visit respectively. These two properties are incremented by appropriate rules formed and applied on the *VisitedThing* instances. Particularly, regarding the visited-assets, a rule evaluates in case there is a *VisitedThing* with OpenClose type and refers to the same system and item id as the *ItemStatistics* respective properties and its asset-id property is different from the "-" value. Similarly, the interesting-assets property is incremented by the same conditions except that the rule evaluates in case the interest-weight property of the *VisitedThing* is over zero. Last, the *ItemStatistics* class incorporates the interest-weight property, which is used to infer interest about the specified item and increases each time one of the following rules evaluates:

- The max-time property of an *ItemStatistics* instance is greater than the typical interaction time specified for the specific item. The typical interaction time is a property, which is explicitly assigned to each *Item* of the system indicating the time needed to explore most of the content of the item.

- The times-visited property of an *ItemStatistics* instance is greater than one and the ratio of total-time/times-visited is above the typical interaction time. This indicates that the visitor not only viewed the specific item many times but also the average time spent was enough (above the typical interaction time) to consider it interesting.

- The ratio of interesting-assets/total-assets of an *ItemStatistics* instance exceeds a specified percent. The visitor has shown interest over a specified percent on the assets an item incorporates.

- The ratio of visited-assets/total-assets of an *ItemStatistics* instance exceeds a specified percent. The visitor has viewed over a specified percent of the assets an item incorporates.

- The percent of the *Item's* tags that appear in the dictionary of tags obtained from the assets visited exceeds a specified limit. In this case, each tag of the *Item* is searched in the dictionary of the obtained tags with appearance-

weight over one. The final percent results from the division of the total-tags-found/item-total-tags.

The *Interests Reasoner* component also infers interest regarding an entire interactive system. In order to model all the required information about a visited system, instances of the *SystemStatistics* class are created. Towards this end, a rule is applied on the *ItemStatistics* instances checking the system id that the instances refer. For each system id discovered, either a new *SystemStatistics* instance is generated in case the system id was found for the first time, or the existing *SystemStatistics* instance is updated. In case a new *SystemStatistics* instance is generated, the system id is assigned to the system-id property of the instance, the visited-items property is assigned the value one and the total-items property is assigned with the number of the items the specific system incorporates. Moreover, the interesting-items property is assigned the value one if the *ItemStatistics* instance for which the rule evaluated was considered interesting from the previous inferences or zero otherwise. Additionally, the interest-weight property of *SystemStatistics* is initialized to zero and increments based on the rules defined for the interest of *SystemStatistics* classes described below. In case that a *SystemStatistics* instance is already generated for the specific system id, its properties are updated. Particularly, the visited-items property is incremented by one and in case the *ItemStatistics* instance that fired the rule has a positive value in its interest-weight property, the interesting-items property of *SystemStatistics* instance is incremented by one.

The rules that infer interest on the visited systems, applied on *SystemStatistics* instances, are the following:

- The ratio of interesting-items/total-items of a *SystemStatistics* instance exceeds a specified percent. The visitor has shown interest in a sufficient percent of the items that the system incorporates.

- The ratio of visited-items/total-items of a *SystemStatistics* instance exceeds a specified percent. The visitor has explored most of the items incorporated into a system.

- The interest-weight property is other than zero both for the current *SystemStatistics* and one created during a past visit of the visitor. This indicates an added interest to a system that the visitor has explored and showed interest in a past visit considering not only the current exhibition but also other exhibition spaces visited.

Finally, the *Interest Reasoner* component by appropriately formed rules generates inferences about the visit in general. There are rules which perform the calculation of the total assets visited from each category, and others that when evaluated inferences about the "smart" exhibits visited are drawn (e.g., how many were visited or regarded interesting etc.). These data are stored in properties of the *Visit* instance in the data-model created for the user exploring the exhibition space. Moreover, in order to draw conclusions about the overall experience of the user, the *Visit* class incorporates the interest-weight property which increments each time one of the rules listed below evaluates:

- The interesting-systems/total-systems of a *Visit* instance exceed a specified percent. The visitor has shown interest in a sufficient percent of the systems incorporated in the exhibition space.

- The visited-systems/total-systems of a *Visit* instance exceed a specified percent. This implies that the visitor has explored most of the systems of the exhibition space.

- The place visited tags match over a percent with the explicitly set interest-tags of the visitor. This indicates that the visitor's interests are close to the exhibited concepts presented in the exhibition space.

- The interest-weight property of the *Visit* instance is greater than zero and there is another *Visit* instance formed during the past regarding the same place and its interest-weight is also greater than zero. This implies that the visitor showed interest about the exhibition during a past visit so an added interest weight is assigned for their current visit.

Once the preprocessing is finished and inferences are generated from the rules described above, the *Main Coordinator* triggers the *Content Manager* component of VisitCollage to extract recommendations based on the interests inferred during this phase. The *Main Coordinator* provides the *Content Manager* the id of the *Visit* instance from which it will not only retrieve the data regarding the items and assets visited but also store the extracted recommendations.

5.7 Content Manager

The *Content Manager* is the component that exploits the information originating from the *Interests Reasoner* in order to discover and deliver personalized recommendations based on the user's interests among the smart exhibits. The *Content Manager* processing consists of two phases:

i. Classification of the *Items* and *Assets* visited based on interest

ii. Generation of recommendations individually for each *Item* and in general the place visited.

The *Content Manager* aims to organize the content into appropriate class instances that the *Presenter* component could easily retrieve from the data-model and appropriately visualize them.

5.7.1 Data Classification

First, the *Content Manager* retrieves the *ItemStatistics* instances, which contain all the information about the items visited. The instances are retrieved into a list structure and ordered descending by the interest-weight property selecting those with interest-weight greater than zero. Thereafter, for each item id in the list the assets visited are discovered. The *Content Manager* with the use of SPARQL queries retrieves from the data-model the assets viewed during the visit through the *VisitedThing* instances formed for the specific item id. The result set includes a list of tuples <asset-id, interest-weight> which are then filtered and stored in

separate lists for each type of asset (images, videos, sounds, texts) and ordered descending by the interest-weight. Furthermore, the *Content Manager* retrieves the tags extracted from the interaction of the user with the assets in a dictionary using as the key the tag's URI and as the value the times the tag appears in order to exploit them for the generation of recommendations.

The *Content Manager* utilizes instances of the *SuggestedThing* class to organize the content, which are stored in the data-model as resources of the *Visit* instance created for the visitor. This class incorporates properties in order to store the classified data described above. These include the **system and item id** of the visited item, the **max-time**, the **times-visited**, the **date-time visited** and the **total-time** spent on this item. All these properties are available from the respective *ItemStatistics* instance of the list previously formed. Moreover, the class incorporates properties where the ids of the asset viewed from each type for this item are stored. Additionally, there are properties where the related items and exhibits are stored as exported during the recommendations generation phase described in the following section.

5.7.2 Recommendations

Once the items visited are retrieved and ordered into the list structure, the process of recommendations extraction follows. During this phase, the algorithm extracts related items and exhibits from the data-model for each item of the list recursively. This is achieved by utilizing both the items of the list and the tags extracted during the interaction of the user with the systems' assets described in the Inferences Generation section.

The first step includes the determination of the tags for which related content will be discovered. Towards this end, both the item's and its interesting assets' tags are retrieved from the data-model and stored in a dictionary with their frequency. Next, the algorithm sequentially checks those tags in order to discover the ones included in the interaction-tags dictionary. Since a tag belongs to both dictionaries, it is added in a list of tuples <tag, frequency> assigned the frequency

indicated in the interaction-tags dictionary. In case the list of tuples is empty, meaning that any of the items-assets tags was found in the interaction-tags dictionary, the list of tuples is populated with the items-assets tags and their frequency. Finally yet importantly, since the entire dictionary is examined, the list of tuples is ordered by frequency in descending order and only the tags' URIs are selected forming the final list of tags. The final list formed from the above filtering contains the most relevant tags for which related content will be discovered. Figure 14 summarizes the method described above through an example.

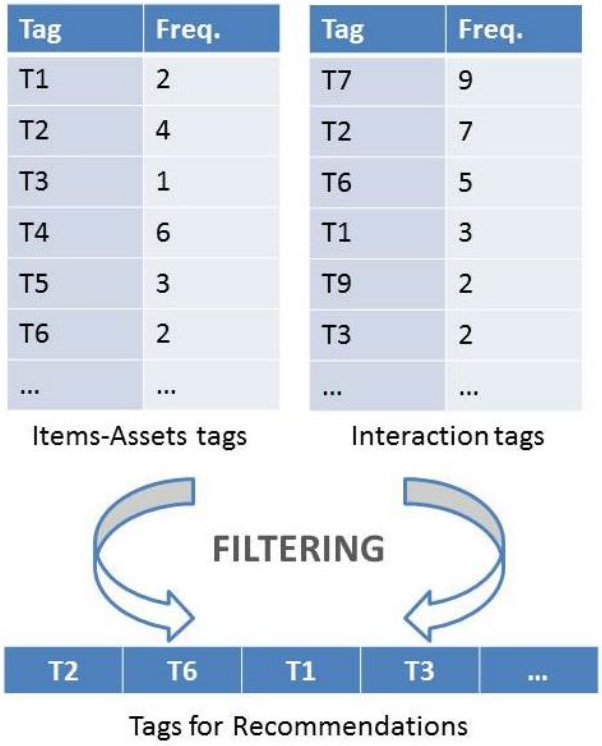


Figure 14: Tags Filtering Process

Next, the algorithm loops over each tag contained in the related-tags list in order to discover items and exhibits in the data-model that incorporate this tag. The discovery process described below for the items is the same for the exhibits. To begin with, for each tag the algorithm generates a list of item-ids that incorporate the specific tag ordered in descending order based on the appearance of the tag either in the item itself or in the assets it incorporates. The algorithm then selects the first item-id from the list that refers to an *Item* that is both not visited by the user and is not already picked to appear in the recommendations. If

selected, the item-id populates the list with the item-ids that will be recommended. This process repeats until either all the tags of the related-tags list are exhausted, or the desired number of recommendations is reached. This number is explicitly declared in the configuration file of VisitCollage.

In case the related-tags list is exhausted and the recommendations number is not reached a further examination of the tags' relations, included in the *Item* instances, takes place. The relations property of the *Item* class stores triples of URIs in the form of <subject, predicate, object> in order to indicate relations of the stored tags' URIs with other resources. Subsequently, the algorithm for each of the tags in the related-tags list obtains either the subject or object resource that the tag participates and populates a list of URIs by which the rest recommendations will be collected. The processing is the same as the one followed with the related-tags list, which is now substituted by the list of URIs formed from the relations of the tags in order to reach the desired number of recommendations for the specific item.

Similarly, the recommended exhibit-ids are discovered based on the related-tags list and in case the desired number is not reached the relations of those tags are used. Once the process terminates both the list with the recommended item-ids and the list with the recommended exhibit-ids are stored in the *SuggestedThing* instance formed for the specific item examined. The recommendations mining is performed recursively for each one of the interesting items forming *SuggestedThing* instances, which carry all the information to be visualized.

The *Content Manager* also extracts recommendations for other exhibition spaces based on the one visited and the interests shown by the visitor during their tour. For this to be achieved, the *Content Manager* utilizes both the tags included in the visited *PlaceOfCulture* instance and the interaction-tags list merging both in a list of tags by which the recommendations are extracted. The rationale for the extraction of related exhibition spaces is the same as the one used for the items and exhibits described above. Once the related exhibition spaces are discovered, their ids are stored in the *Visit* instance of the data-model created for the user.

Finally, the content is organized in the *SuggestedThing* instances, one for each item to be presented. These instances incorporate the information for the item, the assets considered interesting, as well as the recommended items and exhibits for this item. The instances are stored in the data-model and specifically in the *Visit* instance created for the user at the beginning of their tour. Subsequently, since the procedure at this point is terminated, the control passes to the *Main Coordinator*, which in turn notifies the *Presenter* component to visualize the above information.

5.8 Presenter

One of the major characteristics of VisitCollage is the visualization of the monitored information into dynamically generated and personalized web pages. The web-based visit journals created are accessible via a specified URL where the visitors log in using the unique id provided by the reception of the exhibition space.

The web-based visit journals produced are unique for each visitor and tailored to their interaction among the smart exhibits presenting the data mining results: interesting items and recommendations. The *Presenter* component is responsible for the visualization of the content, which currently supports three different kinds of presentation: (i) the sequential, (ii) the spatial and (iii) the booklet.

Particularly, once the interests of the visitor have been extracted and the "interesting" and related content has been generated, the *Main Coordinator* notifies the *Presenter* in order to dynamically create the web pages for the specified *Visit* instance providing the respective visit-id. The *Presenter* in turn using appropriate SPARQL queries retrieves from the ontology the data stored by the *Content Manager* and loads it into internal data structures. The classes used to store the data namely are: (i) the *PlaceInfo*, (ii) the *UserInfo* and (iii) the *VisitInfo.*

The *PlaceInfo* class is used to store all the information about the exhibition space visited (i.e. name, address, description, images etc.), while the *UserInfo* class

instances are populated with data about the user that visited the exhibition (i.e. first and last name, gender, subscription date, email etc.). Last, the *VisitInfo* class is utilized to store information about the visit itself (i.e. title, date of visit, duration, video impressions, etc.) and data extracted during the interaction. These include the *SuggestedThing* instances, related places generated from the *Content Manager* and other statistics such as the number of systems visited or liked, number of images/videos/sounds/texts liked, etc. Moreover, the *VisitInfo* class stores the information about the games the visitor played. The information comes from the *GameAction* class instances stored in the history property of the *Visit* class for the specific user.

Once the data are retrieved and stored into the aforementioned class instances, the *Presenter* proceeds with the generation of the web pages that will constitute the visit journal of the specified visitor. To this end, the *Presenter* incorporates HTML templates specifically developed for each different kind of presentation. Every HTML template includes placeholders, which are populated with actual data before the delivery of the HTML response.

Particularly, the *Presenter* builds the final web page exploiting: (i) the parts that each HTML template consists of, (ii) the CSS file that contains the styles for each element of the template and (iii) the paths to the actual content (i.e. images/videos etc.) which are specified in the configuration file of the system. The parts of each HTML template are maintained in separate files. These files are utilized in the right order, their placeholders are populated with the corresponding data based on the class instances described above and integrated into the main web page. For example, regarding the sequential template, it is separated in an HTML file which includes the markup for the header of the web page, HTML files where the markup for each layout of the presented items is provided, an HTML file with the markup of the layout of the related items etc. Each one of those files includes placeholders at the points where dynamic content will be integrated. The final content is delivered to the location specified in the configuration file of the system and is accessible from the users via a web browser through a specific URL provided at the reception of the exhibition. The following sections present how the

content is visualized for each one of the sequential, spatial and booklet forms of presentation.

5.8.1 Sequential presentation

In this particular form of presentation, the content is presented in a vertical layout. The header of the web page consists of a big image of the visited place along with the user's profile image, their name and information about the visit date and duration. The main section of the web page includes the name of the exhibition space visited at the top, while the interesting items explored by the visitor are laid out sequentially, ordered either by their interest weight or chronologically depending on the user's choice through the respective control.

Each item presented consists of its title, its main image, and its short and long description along with the total time the visitor spent on it during their visit. Moreover, each presented item is followed by three recommendations, which can be either related items or exhibits. Each recommendation consists of a main image, a title and information about its location. Figure 15 shows the layout of the sequential presentation.

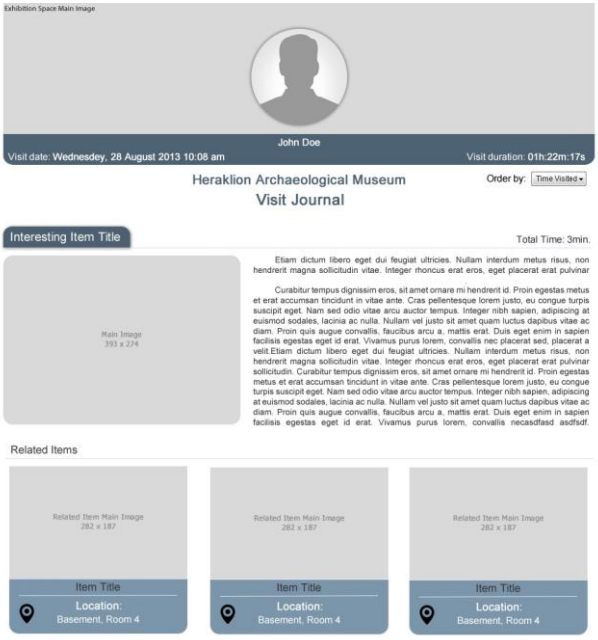


Figure 15: Sequential presentation layout

Depending on the number of the most liked type of assets, the visualization of each item adjusts in order to emphasize on that type. In particular, once the visitor mostly liked images, the layout of the presented item will adjust showing the six most liked images. If the user mostly liked reading texts about an item, the layout adjusts accordingly presenting the two texts of greater interest. Last, if there was a greater preference in videos or sounds, the layout adjusts showing the top three preferred videos or four sounds respectively. The layouts described are presented below.

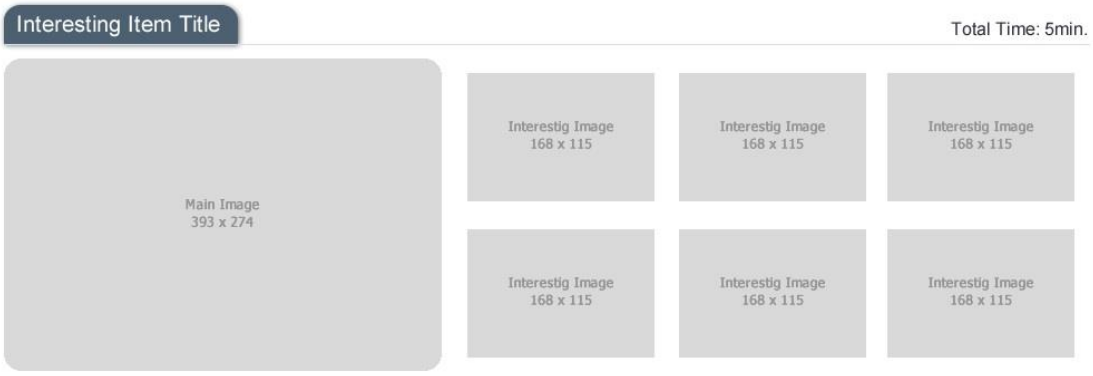


Figure 16: Image-based layout

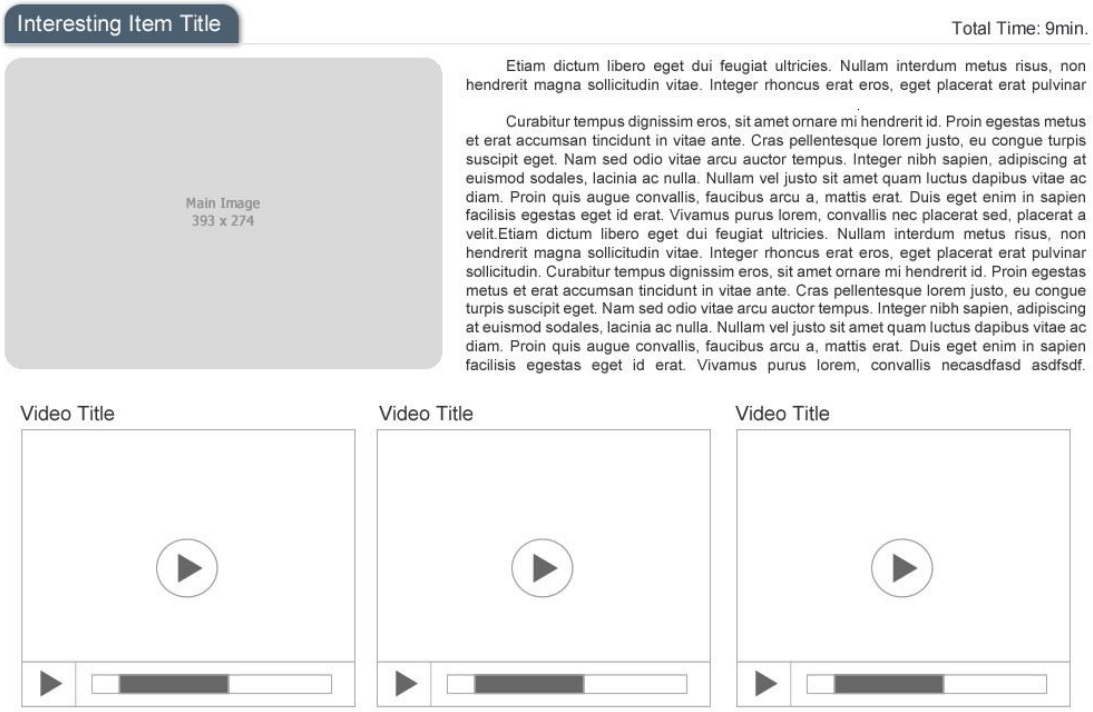


Figure 17: Video-based layout

Figure 18: Text-based layout

Figure 19: Sound-based layout

In case that the visitor played a game during their visit, the layout presented in Figure 20 is used. This layout summarizes the information of the gameplay presenting a main image of the game, the score the visitor achieved and the users the game was played with along with their score. In addition, in case the Game system supports screenshots during the gameplay these are also presented. Lastly,

if a solution of the game is available the user can view it by clicking the respective button.

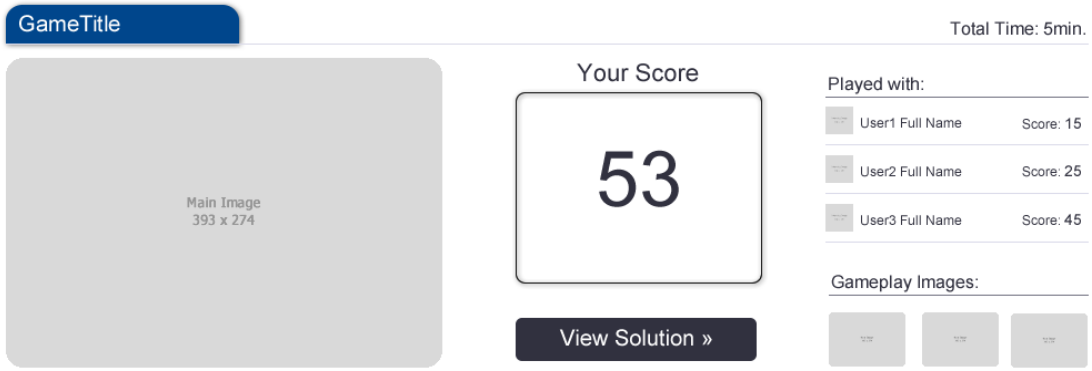


Figure 20: Game layout

The layouts presented in each of the above cases are applied on a specified number of visited items, while the rest are organized into dynamically generated layouts that present an overview of the visited items including their title and main image. An example layout of this form is presented in Figure 21. In order to avoid loading time delays a "load more" button is provided, which when clicked the interface expands showing more items to explore.

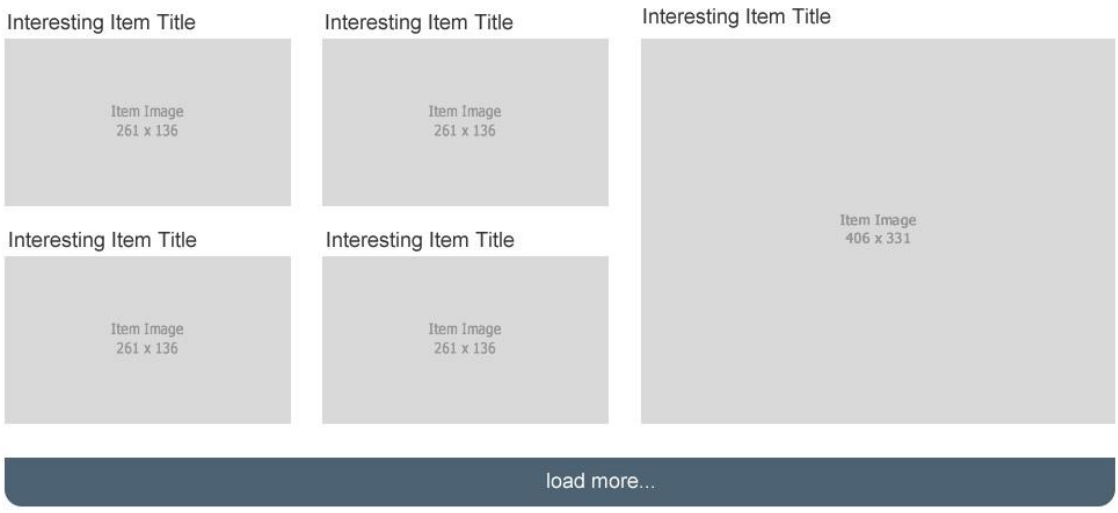


Figure 21: Dynamically generated layout in Sequential presentation

Finally, at the bottom of the interface information about the visited exhibition are presented along with recommendations about similar exhibitions. The

information presented about the visited exhibition includes the title of the exhibition, the short and long description and contact information (i.e. address, phone, website, open hours etc.). In addition, a map is available which depicts the exact location of the exhibition and four photos presenting the exhibition space. The related exhibitions section follows presenting three recommended exhibitions each one consisting of the main image of the exhibition, its title, the short description and a link that leads to the exact location of the exhibition on a map. Figure 22 presents the layout described for the exhibition space visited.

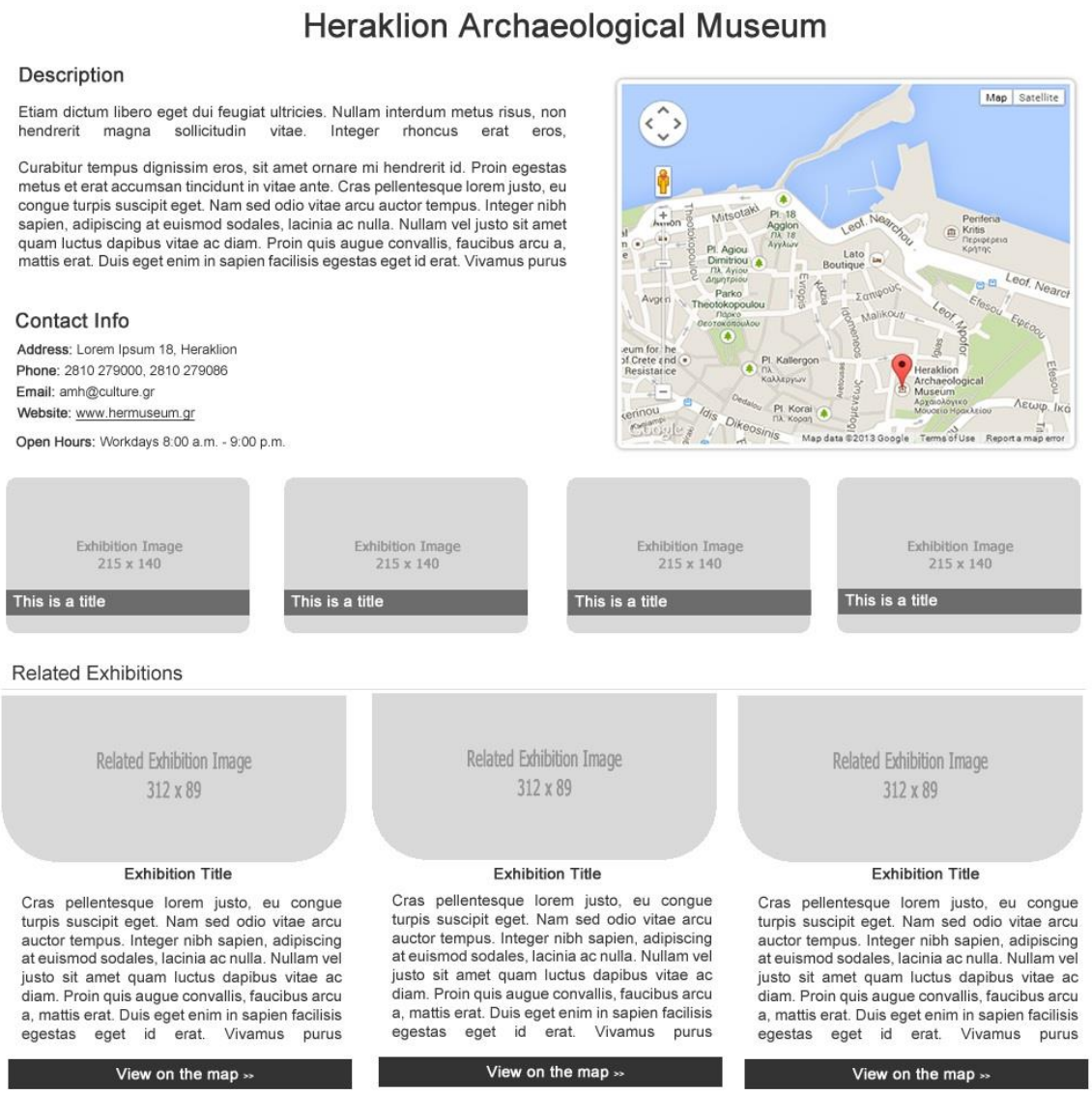


Figure 22: Exhibition layout

5.8.2 Spatial presentation

In this form of presentation, emphasis is given to the location of the visited smart exhibits. The header of the generated web pages consist of the user's full name and profile image on the right side, while on the left side information about the visit date and duration are provided. At the center of the header, the buttons namely "Map" and "Place Info" are provided, by which the user can select to view either the information about the interesting systems and items overlaid on a floor plan of the exhibition or information about the exhibition itself along with recommendations for related exhibitions.

Regarding the first option, the name of the exhibition appears at the top of the main section, followed by its floor plan. The visited interactive systems are arranged within the rooms of the floor plan presenting the most interesting with larger images. The time visited is presented at the bottom of the icon that represents a visited interactive system. Once a visitor selects any of the system-icons, the main images of the items that were inferred interesting are presented into a navigation control right below the floor plan. Moreover, below the navigation control, there is a tabbed menu, which presents the information about the selected item. The "Information" tab of the menu presents the title of the item and its description on the first half, while on the second half assets of this item that were regarded interesting are presented. The "Recommendations" tab presents related items that the visitor can explore followed by their main image, their title and their location similarly to the sequential presentation. The spatial presentation layout described is shown below.

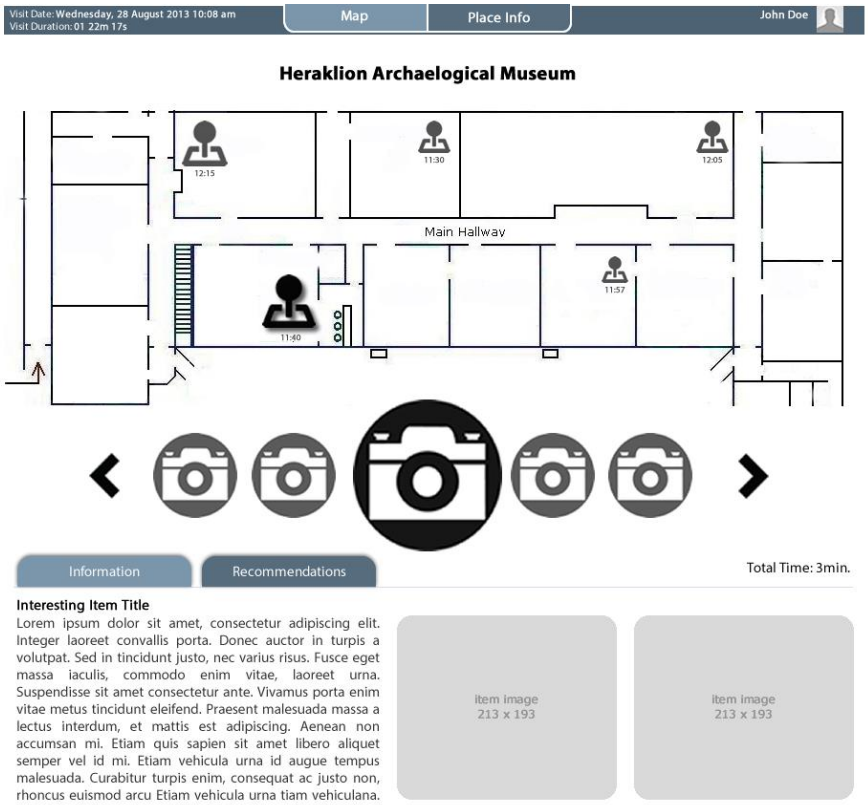


Figure 23: Spatial presentation layout

When the "Place Info" button is selected from the header of the web page, the interface presents information about the visited place in the same way as previously shown in Figure 22.

5.8.3 Booklet presentation

In the booklet form of presentation, the content generated from the user's visit is presented in an interactive booklet where the visitor can navigate through the pages to the interesting items and assets and explore the recommendations provided in a variety of layouts.

Particularly, in this form of presentation, the header of the interface consists of the user's full name and profile image on the right side, while on the left side information about the visit date and duration are provided. At the center of the header, the interface provides options to navigate to the next or previous visit of the user. The main section of the web page consists of the name of the exhibition

followed by the booklet itself. The booklet offers the option to navigate to its pages through two big buttons placed on its left and right side. The content is arranged within the booklet in a variety of layouts. An example layout is presented in Figure 24. In this layout, the left page of the booklet presents the information about an interesting item including a video, its title, its description and three photos arranged below the description. The right page presents the interesting assets of this particular item at the top half; while at the bottom half three related items are provided. The layout that presents the information about the exhibition visited and the related exhibitions is similar to the one presented above. Specifically, the map of the exhibition is placed at the top of the left page, followed by the name of the exhibition, its description and contact information. The right page presents photos from the exhibition at the top half, while at the bottom half three recommended exhibitions are presented.

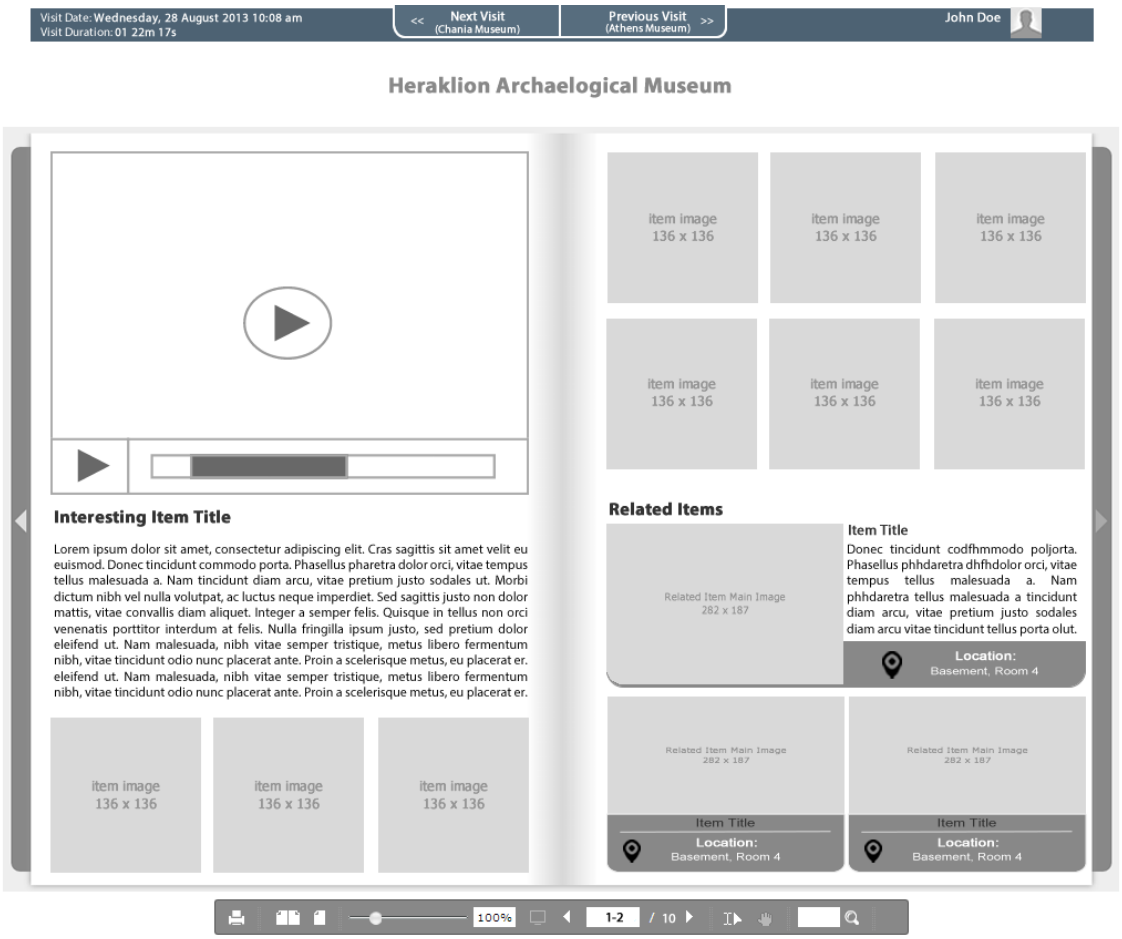


Figure 24: Booklet presentation layout

The footer of the web page provides a list of controls about the booklet. The user has the options to print the booklet, view the booklet page per page, zoom in and out or view the content in full screen. Additionally, the user can navigate to specific pages or even search specific keywords within the booklet.

Chapter 6
Evaluation

6.1 Methodology

As a first step towards the evaluation of the VisitCollage, an expert-based heuristic evaluation was conducted in order to identify usability problems in the design of the three kinds of visualization described in Chapter 5. Heuristic evaluation [50] is the most popular usability inspection method and is carried out as a systematic inspection of a user interface design for usability. This method's goal is to find usability problems in the design so that they can be attended to as part of an iterative design process [51]. The procedure involves having a small set of evaluators examine the interface and judge its compliance with a set of recognized usability principles, namely "heuristics" [52]. In general, heuristic evaluation should involve multiple evaluators to improve the effectiveness of the method, since experience has shown that different people find different usability problems. According to Nielsen [53], the optimal approach is to involve three to five evaluators, since one does not gain much additional information by using larger numbers.

An approach that has been successfully applied is to supply the evaluators with a typical usage scenario, listing the various steps a user would take to perform a sample set of realistic tasks. Afterwards, each individual evaluator inspects the interface alone for usability problems according to the heuristics. The evaluators are allowed to communicate and have their findings aggregated only after all the evaluations have been completed. This procedure is important in order to ensure independent and unbiased evaluations from each evaluator. An observer can be used to assist the evaluators in operating the interface or verbalizing their comments in order to reduce the workload on them. The results of the evaluation are available fairly soon after the last evaluation session since the observer only

needs to understand and organize one set of personal notes, not a set of reports written by others.

Finally, the descriptions are synthesized by the evaluation observer into a list of usability errors along with references to the usability principles that were violated from the aggregate of comments made by the evaluators. Thereafter, each evaluator is called to provide severity ratings on the usability errors in the aggregated list, independently from other evaluators. Severity ratings are used to provide a rough estimate of the need for additional usability efforts. The severity of each usability problem is a combination of four factors: (i) the frequency with which the problem occurs, (ii) the impact of the problem if it occurs, (iii) the persistence of the problem and (iv) the market impact of the problem. The severity ratings range from zero to four and are used to indicate how serious and how important is to fix it.

Table 2 and Table 3 below present the ten usability heuristics and the severity ratings along with their meaning respectively [52].

#	Heuristic	Meaning
1	Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2	Match between system and the real world	The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3	User control and freedom	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo-redo.
4	Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5	Error prevention	Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
6	Recognition rather than recall	Make objects, actions and options visible. The user should not have to remember information from one part of the dialogue to another. Instruction for use of the system should be visible or easily retrievable whenever appropriate.
7	Flexibility and efficiency of use	Accelerators, unseen by the novice user, may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8	Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9	Help users recognize, diagnose, and recover from errors	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10	Help and documentation	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Table 2: The ten usability heuristics

Severity rating	Meaning
0	**I don't agree that this is a usability problem at all**
1	**Cosmetic problem only:** need not be fixed unless extra time is available on project
2	**Minor usability problem:** fixing this should be given low priority
3	**Major usability problem:** important to fix, so should be given high priority

4	**Usability catastrophe:** imperative to fix this before product can be released

Table 3: Severity ratings

Following the aforementioned process, four evaluators based on a typical usage scenario conducted the evaluation of VisitCollage. The scenario described a typical day where a user has visited the Acropolis Museum in which a special exhibition of interactive systems was located. The exhibition consisted of four interactive systems, which the user visited and expressed interest in a number of digital exhibits presented by those systems. At last, upon visit completion, the visitor was given a unique URL in which he could log-in and view the generated journal for his visit.

6.2 Findings

The usability findings identified by the evaluators on the generated visit journals were noted down by an observer, who was monitoring the evaluation progress. The list of the most severe usability issues (average rating over 2.5) along with the violated rules and their mean rating for each presentation alternative are presented below.

Sequential presentation:

Issue #1

Description: Navigation controls to the next and previous visits are missing

Rules violated:
User control and freedom
Consistency and standards
Flexibility and efficiency of use

Severity rating: 3.25

Issue #2

Description: When ordering by interest it is not visible the way interest is measured

Rules violated:
Visibility of system status
Match between system and the real world

Severity rating: 2.65

Issue #3

Description: There is not a way to navigate directly to a specific exhibit

Rules violated:
User control and freedom
Flexibility and efficiency of use

Severity rating: 2.75

Issue #4

Description: There is not a way to return to the top of the page

Rules violated:
User control and freedom
Flexibility and efficiency of use

Severity rating: 2.5

Issue #5

Description: There is not a layout alternative that could support multiple types of multimedia

Rules violated:
Flexibility and efficiency of use

Severity rating: 2.75

Issue #6

Description: It is not clear why the items at the bottom of the page are presented collapsed

Rules violated:
Aesthetic and minimalist design
Recognition rather than recall

Severity rating: 3

Issue #7

Description: It is not clear whether the related items present exhibits that were interesting to the user or exhibits that are semantically related to the current exhibit.

Rules violated:
Recognition rather than recall
Flexibility and efficiency of use

Severity rating: 2.75

Issue #8

Description: It is not clear whether the related items are selectable

Rules violated:
Recognition rather than recall

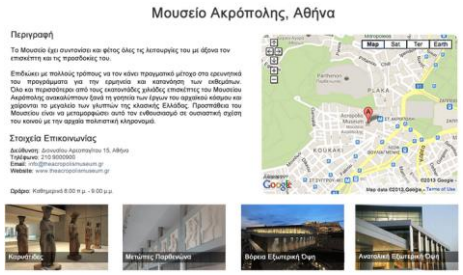
Severity rating: 2.5

Issue #9

Description: It is not clear whether the images presented for place visited are highlights of the recorded visit or of the space in general

Rules violated:
Recognition rather than recall
Aesthetic and minimalist design

Severity rating: 2.75



Issue #10

Description: The suggested exhibitions are not clearly distinguished among the content presented for the exhibition

Rules violated:
Aesthetic and minimalist design
Recognition rather than recall

Severity rating: 2.5

Issue #11

Description: Language inconsistency (English-Greek)
Rules violated:
Consistency and standards
Severity rating: 2.75

Spatial presentation:

Issue #1

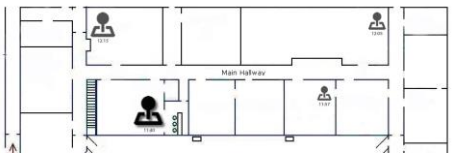
Description: There is not a way to switch among the floors of the exhibition

Rules violated:
User control and freedom
Flexibility and efficiency of use

Severity rating: 3.5

Issue #2

Description: The floor plan does not provide information on how the user navigates among systems

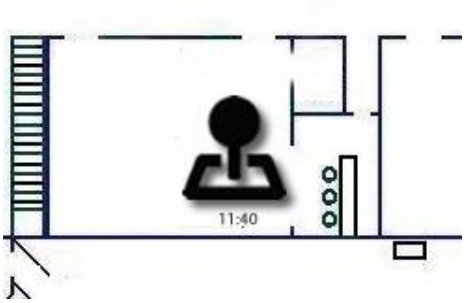


Rules violated:
User control and freedom
Recognition rather than recall

Severity rating: 3.25

Issue #3

Description: Each system presented needs a title



Rules violated:
Recognition rather than recall
Visibility of system status

Severity rating: 2.75

Issue #4

Description: It is not clear whether the selected system is correlated with the central item of the slider

Rules violated:
Visibility of system status
Consistency and standards
Recognition rather than recall
Flexibility and efficiency of use

Severity rating: 3

Issue #5

Description: It is difficult to identify what the slider images represent

Rules violated:
Recognition rather than recall
Visibility of system status

Severity rating: 3.25

Issue #6

Description: Navigation controls to next and previous visits are missing

Rules violated:
User control and freedom
Consistency and standards
Flexibility and efficiency of use

Severity rating: 3.25

Issue #7

Description: Slider images need titles to clarify their purpose

Rules violated:
Aesthetic and minimalist design
Flexibility and efficiency of use
Consistency and standards

Severity rating: 2.5

Issue #8

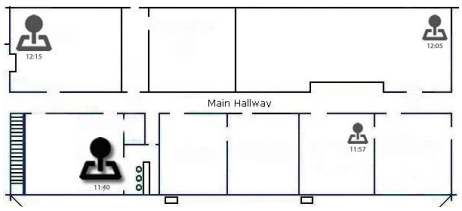
Description: In case there is a lot of content to be presented, the presentation of an exhibit is not scalable

Rules violated:
Flexibility and efficiency of use
Aesthetic and minimalist design

Severity rating: 3

Issue #9

Description: It is not clear why different sizes are used for the pins on the floor plan

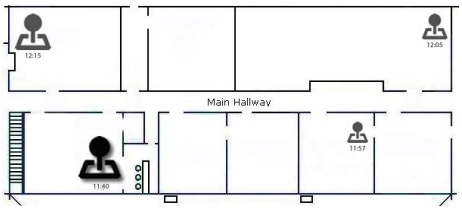


Rules violated:
Visibility of system status
Recognition rather than recall
Flexibility and efficiency of use

Severity rating: 2.5

Issue #10

Description: It is not clear which is the selected pin

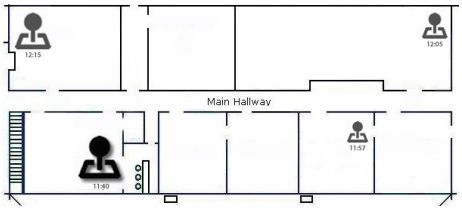


Rules violated:
Recognition rather than recall
Flexibility and efficiency of use
Visibility of system status

Severity rating: 3

Issue #11

Description: The navigation route is not presented on the floor plan

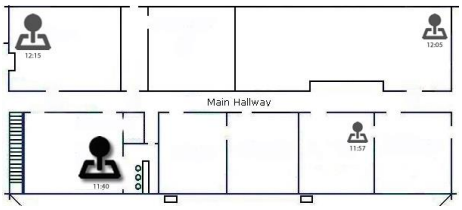


Rules violated:
Flexibility and efficiency of use
Recognition rather than recall

Severity rating: 2.5

Issue #12

Description: There is no differentiation in case there are more than one system in the same room



Rules violated:
Match between system and the real world
Visibility of system status
Recognition rather than recall
Flexibility and efficiency of use

Severity rating: 2.75

Issue #13

Description: There is not a layout alternative that could support multiple types of multimedia

Rules violated:
Flexibility and efficiency of use

Severity rating: 3

Booklet presentation:

Issue #1

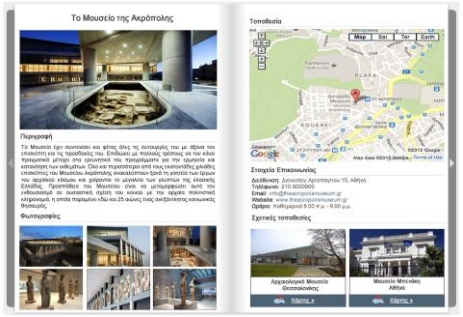
Description: The buttons at the header of the page used for navigating among visits are put conversely

Rules violated:
Match between system and the real world
Error prevention

Severity rating: 2.87

Issue #2

Description: The information presented is not obvious whether it consists selected content or the whole content of the exhibition

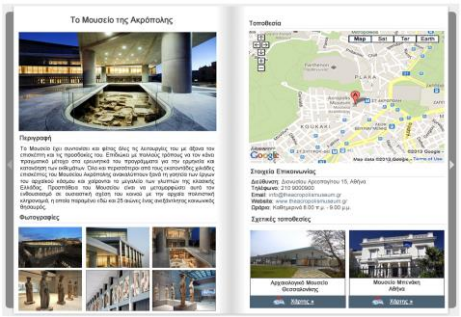


Rules violated:
Visibility of system status
Recognition rather than recall
Aesthetic and minimalist design

Severity rating: 2.75

Issue #3

Description: None layout alternative can support content more than two pages long



Rules violated:
Flexibility and efficiency of use
Aesthetic and minimalist design

Severity rating: 3

Issue #4

Description: It is not clear why the items at the last page are collapsed

Rules violated:
Aesthetic and minimalist design
Recognition rather than recall

Severity rating: 3

Issue #5

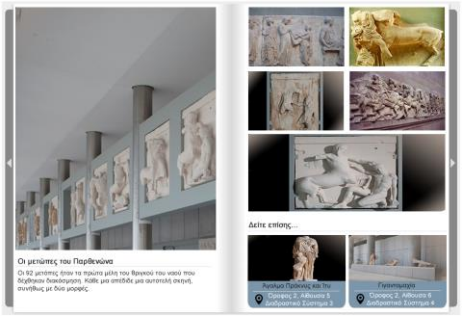
Description: The buttons navigating to the next and previous visits may be considered as the ones handling the pages of the book

Rules violated:
Match between system and the real world
Recognition rather than recall

Severity rating: 3

Issue #6

Description: There is not a layout alternative to present videos

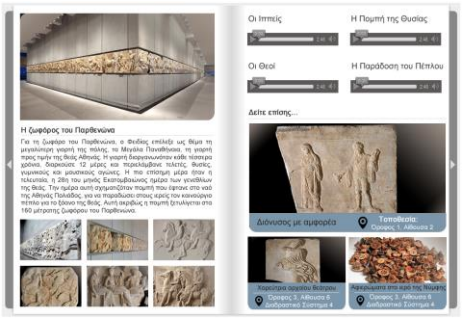


Rules violated:
Aesthetic and minimalist design

Severity rating: 3

Issue #7

Description: It is not clear whether the information of an exhibit can extend to the next page

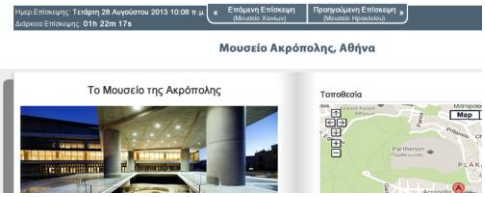


Rules violated:
Recognition rather than recall
Aesthetic and minimalist design
Visibility of system status

Severity rating: 2.75

Issue #8

Description: The header differentiates in comparison to the sequential form of presentation



Rules violated:
Aesthetic and minimalist design
Consistency and standards
Recognition rather than recall

Severity rating: 2.75

Apart from the usability issues described above, the evaluation also revealed several generic issues.

- It is not apparent that the three forms of presentation constitute three alternative ways of presentation among which the visitor can switch (severity rating: 3.5)
- There is not a way to directly navigate to a specific visit without clicking the next or previous button (severity rating: 3)
- Exhibits that the visitor did not explore during the visit are not included in the collage (severity rating: 3)
- No overview that summarizes the visit is provided (e.g., total number of visited exhibits etc.) (severity rating:3)

The findings of the evaluation for each form of presentation are summarized in the following tables and charts.

Sequential presentation

Total usability issues**: 40** Severe issues(rating > 2.5)**: 11**		
Severity Rating	**Total issues**	**Percentage**
[0-1]	3	7.5%
(1-2]	19	47.5%
(2-3]	17	42.5%
(3-4]	1	2.5%

Table 4: Sequential presentation evaluation findings

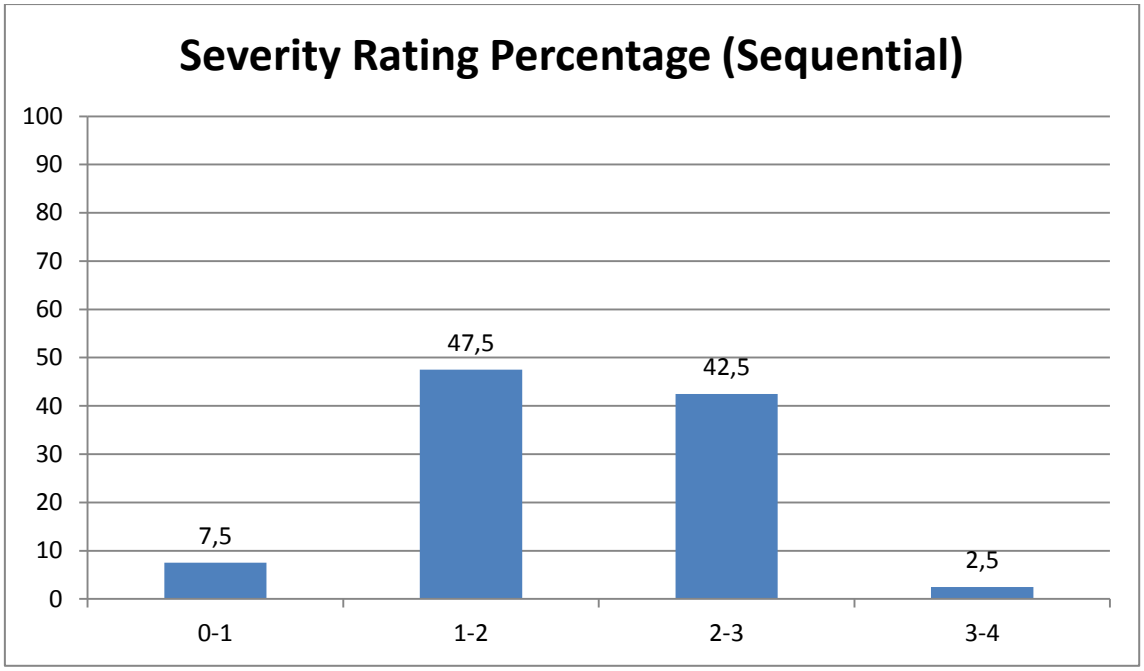


Figure 25: Sequential presentation severity ratings percentage

From the above chart it is clear that most of the usability issues (90%) in the sequential presentation range from one to three. From these problems 47.5% constitute minor usability issues and the 42.5% are major usability issues. These problems mainly violated the "Aesthetic and minimalistic design", "Flexibility and efficiency of use" and "Recognition rather than recall" heuristics. On the contrary, only a few problems were regarded as cosmetic (7.5%) or imperative to fix (2.5%).

Spatial presentation

Total usability issues: **31** Severe issues(rating > 2.5): **13**		
Severity Rating	**Total issues**	**Percentage**
[0-1]	1	3.2%
(1-2]	14	45.2%
(2-3]	12	38.7%
(3-4]	4	12.9%

Table 5: Spatial presentation evaluation findings

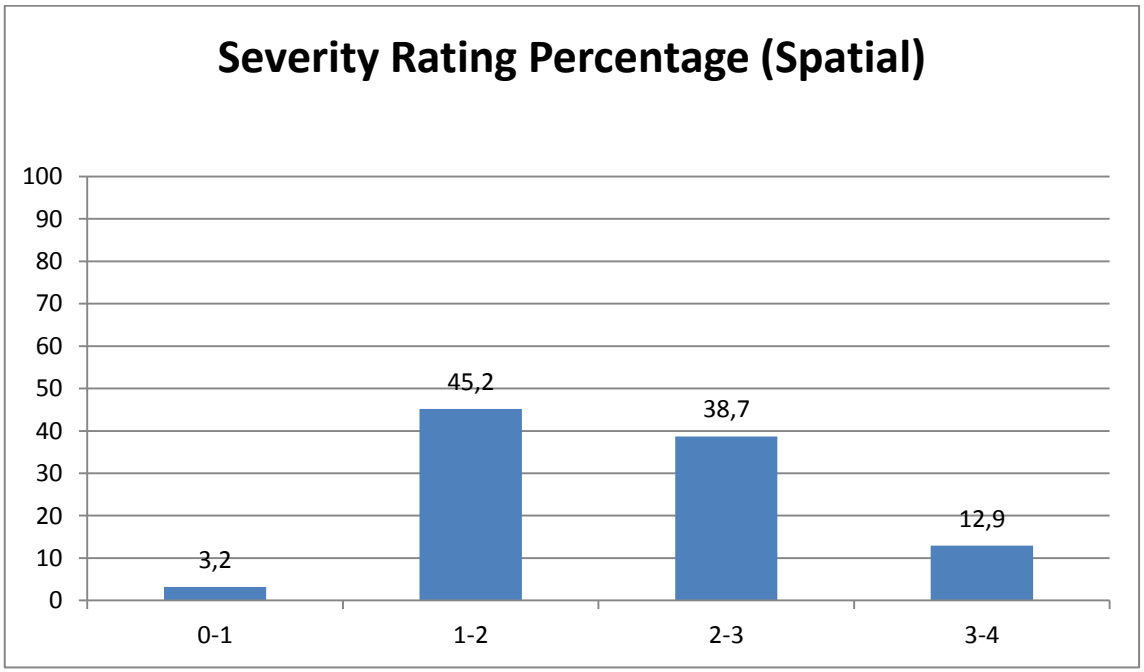


Figure 26: Spatial presentation severity ratings percentage

In the spatial presentation most of the usability problems (45.2 %) range from above one until two, meaning that they are minor usability problems and mainly violated the "Aesthetic and minimalistic design", "Flexibility and efficiency of use" and "Recognition rather than recall" heuristics. There are also problems (38.7%) which constitute major usability problems that should be fixed and mainly

violated the “Flexibility and efficiency of use”, “Recognition rather than recall” and “Visibility of system status” heuristics. Moreover, there are several problems (12.9%) that it is imperative to be fixed mainly violating the “Flexibility and efficiency of use” and “Recognition rather than recall” heuristics, while only a few (3.2%) are regarded of cosmetic nature.

Booklet presentation

Total usability issues**: 27** Severe issues(rating > 2.5)**: 8**		
Severity Rating	**Total issues**	**Percentage**
[0-1]	0	0%
(1-2]	15	55.6%
(2-3]	12	44.4%
(3-4]	0	0%

Table 6: Booklet presentation evaluation findings

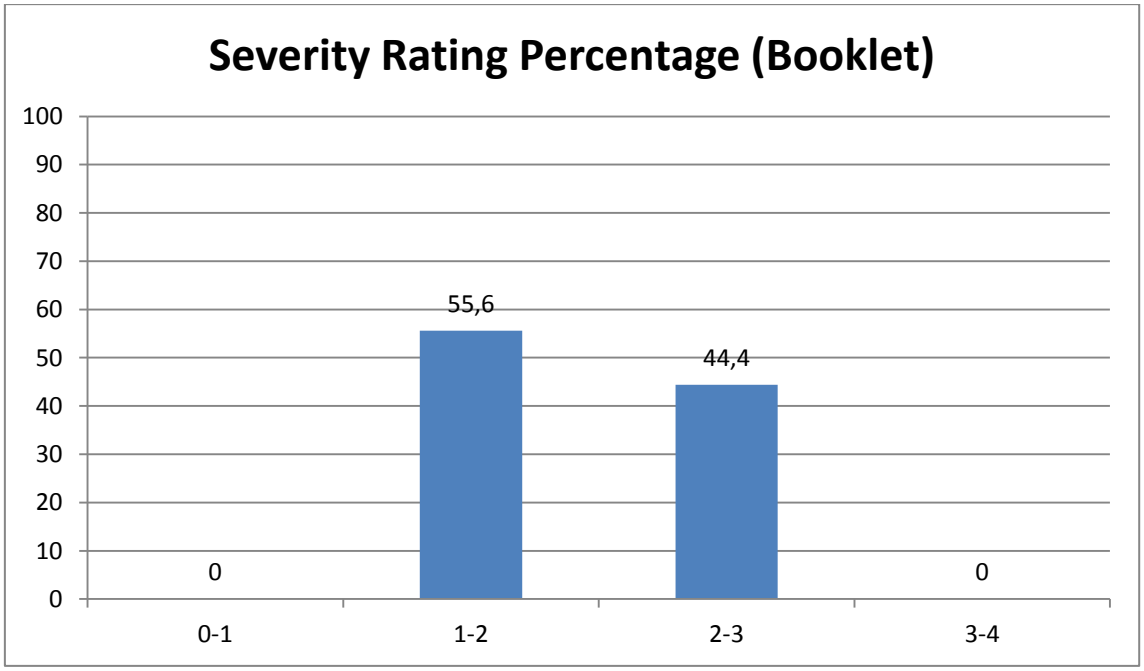


Figure 27: Booklet presentation severity ratings percentage

From the above chart, it is clear that all the severity ratings of the issues in the booklet presentation range from one to three. The 55.6% of the problems constitute minor usability problems while the 44.4% were regarded as major usability problems. The issues identified in both cases mainly violate the “Aesthetic and minimalistic design”, “Flexibility and efficiency of use” and “Recognition rather than recall” heuristics. On the contrary, any of the problems discovered were rated as cosmetic or a usability catastrophe.

In general, as presented in Figure 28 the sequential presentation seems to have the most usability issues (40), while only 11 were rated as severe (average rating > 2.5), which is translated to 27.5% of the total problems. Regarding the spatial presentation, 13 out of 31 total issues were rated as severe which means 42% of the total problems. Last, in the booklet presentation the evaluators found 27 total issues whereas only 8 were rated as severe, which is translated to 29.5% of the total issues. From the aforementioned percentages, it can be assumed that the spatial presentation is the most problematic form of presentation. This is also supported by the average rating of the usability issues in this particular form (2.125 out of 4) in contrast to the other two forms (2 out of 4) as presented in Figure 29.

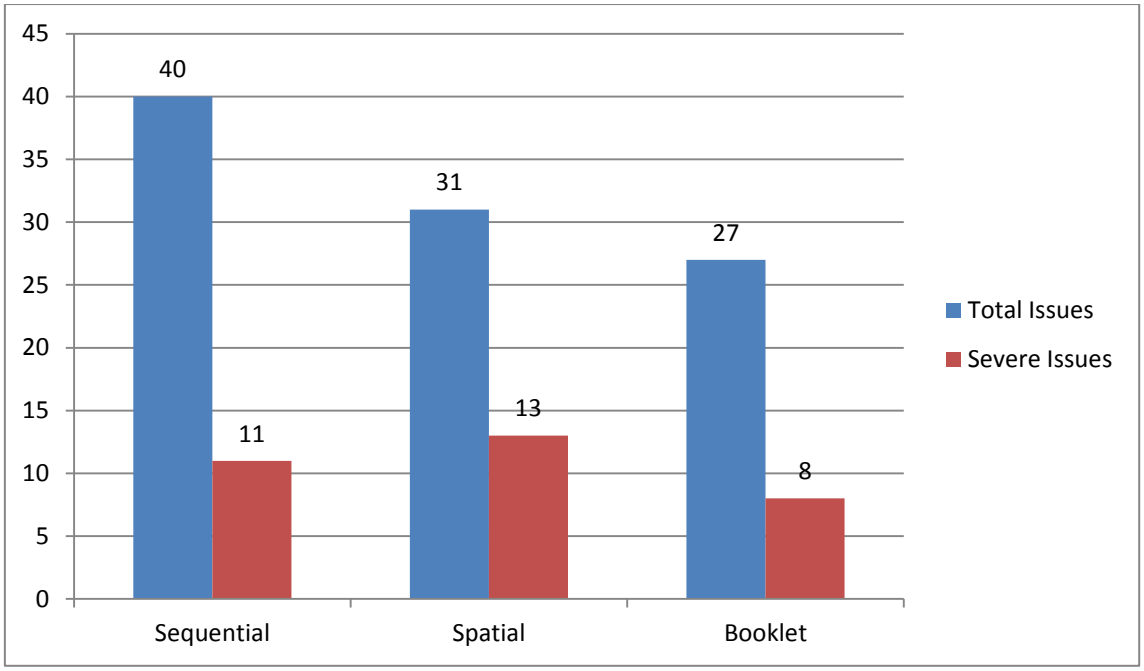


Figure 28: Total vs. Severe usability issues

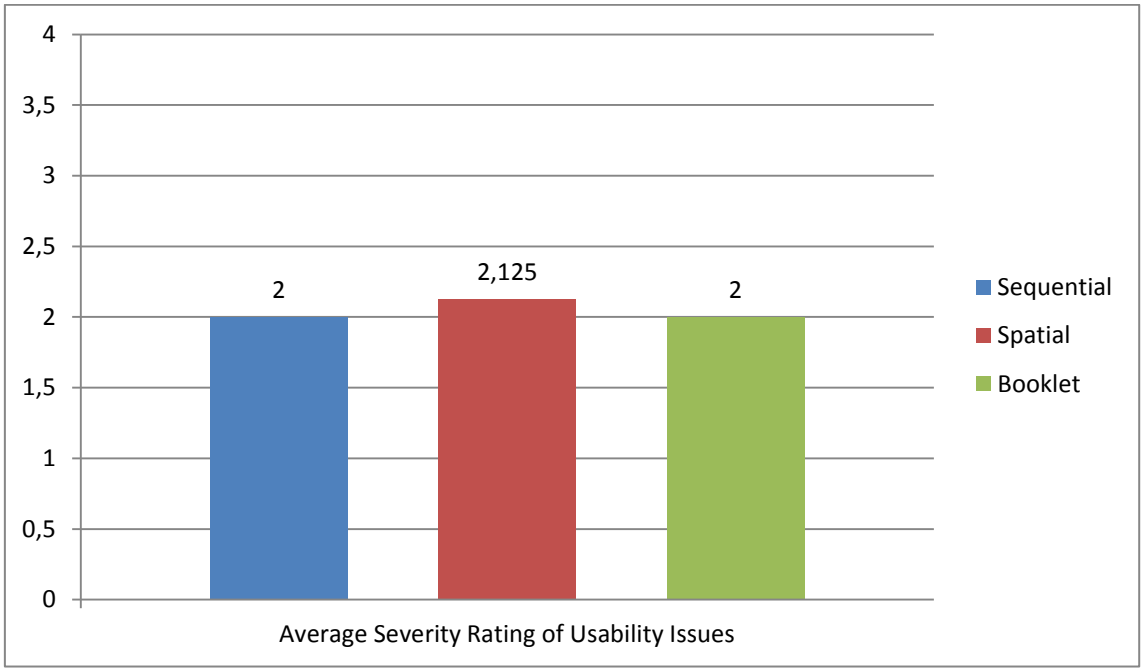


Figure 29: Average severity rating of usability issues

Chapter 7
Conclusion and future work

7.1 Summarizing

Ambient Intelligence is an emerging field of research that promises to enhance traditional user interfaces and augment information visualization in smart environments, where users can interact in a natural manner. Many exhibition spaces are increasingly augmenting the space with digital installations, exploiting the evolution of ubiquitous technologies. Therefore, the need for systems that can enhance the visitors' experience and engagement to the exhibitions is emerging. The VisitCollage system proposed in this thesis aims to cover the gap between those installations and the visitors' exploration in order to augment their experiences.

The VisitCollage provides a significant contribution towards enhancing visitors' experience in a ubiquitous exhibition space as it addresses all the aspects of an exhibition visit from monitoring the activities of the visitors to the delivery of personalized content through web-based, dynamically generated visit journals. In particular, the VisitCollage is built upon a modular architecture. The VisitCollage's core consists of six major components: the *Systems Manager*, the *Users Manager*, the *Actions Logger,* the *Interests Reasoner,* the *Content Manager* and the *Presenter* all of which are coordinated by one last component called *Main Coordinator.*

Specifically, the *Systems Manager* enables the insertion of data into the system that describe attributes of the exhibition space, the systems exhibited and all the information that these systems incorporate. The *Users Manager* as indicated by its name is the component that handles the data regarding the users of the system and their profiles. Once a visitor starts exploring the smart exhibits, the *Actions Logger* component of VisitCollage is responsible for logging visitors' interaction through appropriate action services provided by VisitCollage to the smart exhibits to communicate. When a visitor terminates their exploration, the

Interests Reasoner undertakes the process to generate inferences about the visitors' interests based on their activity among the smart exhibits. At this phase, a two-step procedure is followed and includes the preprocessing of the activity and the reasoning. For the reasoning, the Euler engine is used exploiting a set of rules along the preprocessed data in order to generate inferences about the visitors' interest. The *Content Manager* then exploits the information originating from the *Interests Reasoner* and by appropriate filtering methods delivers personalized recommendations tailored to the visitors' interests.

Finally, the *Presenter* component of VisitCollage is responsible for the visualization of the interests and recommendations extracted for the users. The *Presenter* reconstructs the captured experience into personalized, dynamically generated web pages, which the visitor can explore and share through social networks while either on-site or on-line. The dynamically generated web content is presented in three different forms: (i) the sequential, (ii) the spatial and (iii) the booklet, aiming to enrich the visualization, increase visitors' satisfaction, and provide them motivation for multiple visits.

Overall, it can be claimed that the work presented in this thesis constitutes a significant step towards supporting the extensive use of AmI technologies in the context of exhibition spaces for augmenting visitors' experience and engagement.

7.2 Future work

Regarding future work, additional steps could be taken to fully support the described concept of VisitCollage. The next step of this work would be to augment the available content and conduct an exhaustive user-based evaluation in order to acquire additional feedback from end users. Moreover, another extensive evaluation should take place to validate the rules' accuracy and efficiency. The results of both evaluations would lead to further improvements of VisitCollage in order to meet the visitors' needs.

Additionally, some relevant topics are currently investigated for future upgrades. In terms of context, VisitCollage can integrate additional information about the context in order to improve recommendations. For example, the system could exploit the permanent location of the visitor in order to suggest nearby exhibitions or even consider the timetable of the attractions to limit the suggestions to the currently open. Moreover, weather data could be exploited during the inference of recommendations in order to suggest exhibitions that are indoor or outdoor respectively.

Regarding the visitors' profile, the system could afford the connection with social networks (e.g., Facebook) for making an intelligent estimate of the users' context and give the ability to users to change explicitly any incorrect information or assumptions made for personalization purposes. Another interesting issue would be the potential to support monitoring of physical exhibits through appropriate services as the system provides the infrastructure to store such information.

Another great extension of the system would be to provide even more ways of visualization by extending the *Presenter* component and provide mechanisms to feed other applications (e.g., Timelines [54]) installed in the exhibition space in order to augment the on-site experience of the visitors. Another extension on the visualized content would be to provide mechanisms to monitor visitors' activity while viewing the generated visit journals thus forming a meta-system, which would provide further feedback to VisitCollage about the visitors' interests.

Finally, supplementary graphical tools could be created in order to facilitate content insertion, rules editing, templates creation, and exhibition space management. Particularly, regarding the content, a CMS (i.e., Content Management System) could be developed in order to provide the ability to insert, update or delete content from VisitCollage through an easy-to-use graphical environment. The rules editor would also be a beneficial tool towards the creation of new rules or adjusting the implemented Notation3 rules. Ideally, entities, properties and values from the data-model will be presented as graphical modules, which by

simple drag and drop could build new rules or adjust the existing ones. As far as visualization is concerned, a graphical tool that offers the ability to build new templates for the *Presenter* component would surely constitute an important extension. Eventually, a management tool that would exploit the tracked information of the visitors in order to understand their behaviour and preferences and shape evaluation metrics would be an added value for the exhibition management. Example functions of such tool could be measurement of the popularity of exhibits, sequences of interactions that tend to appear more frequently in order to define trends in the exhibition space that could be used to reposition the exhibits or design effective campaigns for the future.

Bibliography

[1] I. (. A. Group), *Ambient Intelligence: from vision to reality,* 2003.

[2] M. Weiser, *The Computer for the 21st Century,* 1991.

[3] M. Maybury, "Intelligent user interfaces: an introduction," in *Proceedings of the 4th international conference on Intelligent user interfaces*, 1998.

[4] J. Bowen and S. Filippini-Fantoni, "Personalization and the web from a museum perspective," in *Museums and the Web*, 2004.

[5] T. Erickson, "Designing visualizations of social activity: six claims," in *CHI EA '03 CHI '03 extended abstracts on Human factors in computing systems*, 2003.

[6] J. Hitzeman, C. Mellish and J. Oberlander, "Dynamic Generation of Museum Web Pages: The Intelligent Labelling Explorer," *Archives and Museum Informatics,* vol. 11, pp. 107-115, 1997.

[7] I. Mathes, A. Pateli, A. Tsamakos and D. Spinellis, "Context aware services in an Exhibition Environment-the mEXPRESS approach," *Proceedings eBusiness and E-work Conference,* 2002.

[8] O. Stock, M. Zancanaro, P. Busetta, C. Callaway, A. Krüger, M. Kruppa, T. Kuflik, E. Not and C. Rocchi, "Adaptive, intelligent presentation of information for the museum visitor in PEACH," *User Modeling and User-Adapted Interaction,* vol. 17, no. 3, pp. 257-304, 2007.

[9] C. Rocchi, O. Stock, M. Zancanaro, A. Krüger and M. Kruppa, "The Museum Visit : Generating Seamless Personalized Presentations on Multiple Devices," *Computer,* pp. 316-318, 2004.

[10] S. Hsi and H. Fait, "RFID enhances visitors' museum experience at the Exploratorium," *Communications of the ACM,* vol. 48, pp. 60-65, 2005.

[11] S. Hsi, R. Semper, W. Brunette, A. Rea and G. Borriello, "eXspot : A Wireless RFID Transceiver for Recording and Extending Museum Visits," pp. 2-3.

[12] M. Fleck, M. Frid, T. Kindberg, E. O'Brien-Strain, R. Rajani and M. Spasojevic, "From informing to remembering: Ubiquitous systems in interactive museums," *Ieee Pervasive Computing,* vol. 1, pp. 13-21, 2002.

[13] M. Fleck, M. Frid, T. Kindberg, E. O'Brien-Strain, R. Rajani and M. Spasojevic, "Rememberer : A Tool for Capturing Museum Visits," pp. 48-55, 2002.

[14] Z. Yu, X. Zhou, Z. Yu, J. Park and J. Ma, "iMuseum: A scalable context-aware intelligent museum system," *Computer Communications,* vol. 31, no. 18, pp. 4376-4382, 2008.

[15] R. Baker, G. Roussos and M. Levene, "The Experience Re-player: Trail-based Reconstruction of Captured Experiences in Ubiquitous Computing Spaces," *dcs.bbk.ac.uk.*

[16] L. Pujol, M. Roussou, S. Poulou, O. Balet, M. Vayanou and Y. Ioannidis, "Personalizing interactive digital storytelling in archaeological museums: the CHESS project," *CAA 2012,* 2011.

[17] I. Liiv, T. Tammer, T. Ruotsalo and A. Kuusik, "Personalized Context-Aware Recommendations in SMARTMUSEUM: Combining Semantics with Statistics," *2009 Third International Conference on Advances in Semantic Processing,* pp. 50-55, 2009.

[18] A. Luberg, T. Tammet and P. Järv, "Smart City: A Rule-based Tourist Recommendation System," *Information and Communication Technologies in Tourism 2011,* 2011.

[19] E. Hornecker and M. Stifter, "Digital backpacking in the museum with a SmartCard," *Proceedings of the 6th ACM SIGCHI New Zealand chapters international conference on Computerhuman interaction design centered HCI CHINZ 06,* no. July, pp. 99-107, 2006.

[20] Z. Yu, D. Zhang, X. Zhou and C. Li, "User preference learning for multimedia personalization in pervasive computing environment," *Knowledge-Based Intelligent Information and Engineering Systems,* pp. 236-242, 2005.

[21] M. Hatala, R. Wakkary and L. Kalantari, "Rules and ontologies in support of real-time ubiquitous application," *Web Semantics Science Services and Agents on the World Wide Web,* vol. 3, pp. 5-22, 2005.

[22] Y. Wang, N. Stash, L. Aroyo, P. Gorgels, L. Rutledge and G. Schreiber, "Recommendations based on semantically enriched museum collections," no. Section 6.

[23] Y. Wang, L. Aroyo, N. Stash, R. Sambeek and P. Gorgelis, "Cultivating personalized museum tours online and on-site," *Interdisciplinary Science Reviews,* vol. 34, pp. 139-153, 2009.

[24] L. Rutledge, L. Aroyo and N. Stash, "Interactive user profiling in semantically annotated museum collections," *Proc. 5th Int. Semantic Web Conference,* pp. 2-4, 2006.

[25] L. Rutlledge, L. Aroyo and N. Stash, "Determining user interests about museum collections," *Proceedings of the 15th international conference on World Wide Web - WWW '06,* p. 855, 2006.

[26] F. Cena, F. Carmagnola, O. Cortassa, C. Gena, Y. Wang, N. Stash and L. Aroyo, "Tag Interoperability in Cultural Web-based Applications," pp. 5-6.

[27] R. Karimi, A. Nanopoulos and L. Schmidt-thieme, "RFID-Enhanced Museum for Interactive Experience," *Multimedia for Cultural Heritage,* pp. 192-205,

2012.

[28] K. Meehan, T. Lunney, K. Curran and A. McCaughey, "VISIT: Virtual Intelligent System for Informing Tourists," *PGNET, Liverpool,* 2012.

[29] K. Meehan, T. Lunney, K. Curran and A. McCaughey, "Context-Aware Intelligent Recommendation System for Tourism," *scis.ulster.ac.uk,* no. 3, 2013.

[30] M. Telem, "Information requirements specification I: Brainstorming collective decision-making approach," *Information processing & management,* vol. 24, no. 5, pp. 549-557, 1988.

[31] "Macedonia from fragments to pixels," [Online]. Available: http://www.makedonopixels.org/.

[32] W3C, "Resource Description Framework (RDF)," 2004. [Online]. Available: http://www.w3.org/RDF/.

[33] R. V. Guha and D. Brickley, "RDF Vocabulary Description Language 1.0: RDF Schema," 10 February 2004. [Online]. Available: http://www.w3.org/TR/rdf-schema/.

[34] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," 2008.

[35] J. D. ROO, *Euler proof mechanism,* 2005.

[36] T. Berners-Lee, "Notation 3-An readable language for data on the Web," 2006.

[37] "Protégé Ontology Editor and Knowledge-base Framework Overview," [Online]. Available: http://protege.stanford.edu/.

[38] M. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson and N. Noy,

"Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé," in *Workshop on Interactive Tools for Knowledge Capture (K-CAP-2001)*, 2001.

[39] J. Tauberer, "SemWeb.NET: Semantic Web/RDF Library for C#/.NET," 2010. [Online]. Available: http://semanticweb.org/wiki/SemWeb-DotNet.

[40] "Euler Proof Mechanism," [Online]. Available: http://www.agfa.com/w3c/euler/.

[41] Georgalis, Y., Grammenos, D., & Stephanidis, C. (2009). Middleware for Ambient Intelligence Environments: Reviewing Requirements and Communication Technologies. In C. Stephanidis, (Ed.), *Universal Access in Human-Computer Interaction - Intelligent and Ubiquitous Interaction Environments. - Volume 6 of the Proceedings of the 13th International Conference on Human-Computer Interaction (HCI International 2009)*, San Diego, CA, USA, 19-24 July, pp. 168-177. Berlin Heidelberg: Lecture Notes in Computer Science Series of Springer

[42] W3C, "HTML5," World Wide Web Consortium, WHATWG, December 2012. [Online]. Available: http://www.w3.org/TR/html5/.

[43] Netscape Communications Corporation, Mozilla Foundation, 1995. [Online]. Available: http://en.wikipedia.org/wiki/JavaScript.

[44] W3C, "Cascading Style Sheets," World Wide Web Consortium, 17 December 1996. [Online]. Available: http://www.w3.org/Style/CSS/.

[45] E. Gamma, Design patterns: elements of reusable object-oriented software, Addison-Wesley Professional, 1995.

[46] D. Brickley and L. Miller, "FOAF vocabulary specification 0.91," 2000.

[47] M. Doerr, "The CIDOC CRM – an Ontological Approach to Semantic

Interoperability of Metadata," *AI Magazine,* October 2001.

[48] "The CIDOC Conceptual Reference Model," [Online]. Available: http://www.cidoc-crm.org/.

[49] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Stopwords.

[50] J. Nielsen, J. Molich and R. Molich, "Heuristic evaluation of user interfaces," in *SIGCHI conference on Human factors in computing systems*, 1990.

[51] J. Nielsen, "Finding usability problems through heuristic evaluation," in *Proceedings of the SIGCHI conference on Human*, 1992.

[52] J. Nielsen, "Usability Inspection Methods Jakob Nielsen," in *Conference companion on Human factors in computing systems*, 1994.

[53] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, 1993.

[54] G. Drossis, D. Grammenos, I. Adami and C. Stephanidis, "3D Visualization and Multimodal Interaction with Temporal Information Using Timelines," in *Human-Computer Interaction – INTERACT 2013*, 2013.