The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.104 e-ISSN: 2251-7545

A Dynamic Rule-based Approach for Self-adaptive Map Personalisation Services

Basel Magableh
School of Computer Science and Informatics,
University College Dublin
Dublin, Ireland
basel.magableh@ucd.ie

Michela Bertolotto
School of Computer Science and Informatics,
University College Dublin
Dublin, Ireland
michela.bertolotto@ucd.i

Abstract* - With continuous increase of available geographical information services, requirements for personalising map content according to the user's profile and context information are increasingly important. Map personalisation applications could adapt their functionality/behaviour to provide the user with specific spatial data related to his interests at runtime. However, this can be achieved if map applications are able to filter and prioritise geospatial data using dynamic decision-making processes, which considers users' profiles and context for selecting and styling map content according to their needs. To this aim, this article proposes a new approach for map personalisation using dynamic rule-based engine, which provides the map application with the ability to change its styles and rules dynamically according to users' profiles. This approach differs from the majority of existing works, which seek to embed the styling rules on the functional implementation of the map application. In addition, the personalisation engine is integrated with context-driven adaptation, which allows the application to monitor, detect, analyse, and react over changes on the computational environment and users' profiles. This enables map applications to use a styling rule that provides different levels of personalisation and adapt to changes in the computational environment including level of resources and quality of services.

***Index Terms*—self-adaptive map personalisation service, con-text oriented software development, map personalisation**

I. INTRODUCTION

GIS software can provide an effective mechanism for collecting, analysing and displaying geographic information to different individuals, including end users, geo-scientists and decision makers [1]. The quantity of available detailed spatial content that is displayed on a digital map leads to the problems of information overload, lengthy map downloading and rendering time. A reasonable approach to overcome these problems is to provide an effective and unobtrusive mechanism that implicitly filters and prioritises geospatial content based on the user's profile. Such process is called map personalisation. Personalisation, in the context of this article, refers to the ability of map software systems to adapt geospatial content based on the user's profile and interests.

Personalisation techniques have been designed using pre-defined rules and conditional expressions for personalising map content according to user's profile. Unfortunately, using pre-defined rules and conditional expressions often leads to poor maintainability and scalability as changing these rules when needed requires intervention of the software developers. In addition, many personalisation techniques [1], [2], [3] have considered a client-server model on their software architecture design, which often faces lengthy map downloading and long rendering time, as all personalisation is done at the server side of the architecture [4]. Moreover, most personalisation techniques (e.g. [5], [6], [7]) have been proposed without considering the computational context of the environments, which often leads to poor adaptability to the variability of resources and services.

Map personalisation applications could adapt their functionality/behaviour to provide the user with a specific spatial data related to his/her interest at runtime. This functionality needs map software to have the ability of self-adaptability and context-awareness. Context-awareness refers to the software ability to monitor and detect contextual changes in the environment where they operate. In general, context can be defined as any information that is computationally accessible and upon which application behaviour depend [8]. The context is classified based on its type and whether it comes from the computational environment or the user profile. A computational context can be provided by a physical or logical source, i.e. available memory, or resources, i.e. battery power level and bandwidth. The profile includes the user's personal in-formation and interests based on his/her historical map usage. Self-adaptability refers to the software ability to adjust its own functionality or behaviour in response to contextual changes [9]. Enabling GIS software with self-adaptability and context-awareness can increase the level of support they provide to end-users and their ability to adjust their functionality and behaviour dynamically [10]. In this regard, self-adaptive map personalisation service refers to a class of software that can analyse and display geospatial data based on context changes within the environment where they operate. In addition to that, they can modify map content and display required amount of geospatial data according to the user's profile and interests. Such dynamic personalisation service requires a dynamic-decision making engine, which can propose a precise styling rule for each specific condition.

To this aim, this article proposes the use of a dynamic rule-based personalisation engine for customising map content. The personalisation engine is implemented at the client side and provides the map application with the ability to change its styles and rules dynamically according to the users' profiles. The main advantage of this personalisation engine is the ability to filter and prioritise geospatial data using dynamic decision-making processes, which considers users' profiles for selecting and styling map content according to their needs. In addition, the personalisation engine is integrated with context-driven adaptation techniques, which allow the application to moni-tor, detect, analyse, and react over contextual changes. This provides map applications with the ability to adapt to changes in the computational environment and provide different levels of personalisation for users.

Implementing a dynamic rule-based personalisation engine at the client side of the architecture provides the map ap-plication with the following benefits: 1) the personalisation engine can update the rules' syntax based on the evolution of the user's profile and his/her interest without changing the map application's code. 2) Separating the rules' code from the application code requires less intervention from the software developers and reduces the maintainability efforts. 3) Allow the map application to offer different levels of personalisation for different users, as it can handle a wide range of users' profiles without encoding their rules on the map application's code. 4) Increase the efficiency of personalisation by sending specific personalisation requests to the Web Mapping Service (WMS). This allows the map application to request a specific set of map features from the WMS that requires less computational capacity like network bandwidth. 5) Preforming the personalisation at the client side overcomes the problem of information overload and lengthy map downloading found in server-side personalisation.

The rest of the article is structured as follows. Section II briefly sketches the technology used for achieving map personalisation. Section III proposes the mechanism of context-driven personalisation. Section IV illustrates a case study implemented by Context Oriented Software Development (COSD) methodology. The development process and the design of self-adaptive map personalisation services, including the self-adaptive map application and the personalisation engine, are described in details in Section V. Section VI focuses on evaluating the performance of the personalisation process using the dynamic rule-based engine against the static approach of personalising map content.

II. RELATED WORK

Typically, map personalisation systems use a client-server model for customising map content. CoMPASS [11], RecoMap [5] and GeminiMap [3] are examples of applications that pro-vide map personalisation based on users' profile. CoMPASS personalises the geospatial dataset by adding or removing features from the map based on user's profiles. GeminiMap [3] personalises the visual appearance of the spatial content by highlighting map features. These personalisation systems implicitly profile the user's interests using statistical modelling techniques, which provide an interest score for each user. Interest scores have been used extensively in the non-spatial domain, particularly in web site browsing as discussed by
Claypool et al. [12]. Ballatore et al. [5] proposed RecoMap as a recommendation system for map users. RecoMap identifies user interests by monitoring user interaction and context to provide recommendations associated to map features. Re-coMap uses a separated component for sensing the users' context and profiling their interaction with map. Based on that it assigns interest scores to spatial items by combining user's interactions and proximity with a specific map feature. CoMPASS, RecoMap and GeminiMap use an implicit profiling technique of users' interactions with map applications, which allows them to perform the personalisation at the server side. However, these personalisation techniques have not considered the computational context of the environments, which often leads to poor adaptability to the variability of resources and services.

Providing dynamic and adaptive map personalisation re-quires the service to be able to adjust, filter, or prioritise map features based on the execution context of the user and his/her profile. This is only achieved if the service is designed to be dynamic and offers adaptability in conjunction with the variability of the computational environments in which the map application operate. In this sense, map personalisation can be achieved using External or Internal adaptation. The internal approach encodes the adaptation action in the application logic. This approach is based on programming language techniques such as conditional expressions, parametrisation and exceptions [9], [13]. In the internal approach, the whole set of sensors interface, context model, user's profile, and adaptation processes are embedded within the application code, which often leads to poor scalability and maintainability. A slight change in the context model requires intensive code maintenance for the map application code. Based on that, the personalisation techniques proposed in [5], [6], [7] can be considered as internal personalisation techniques that encode the map styling rules internally in the application code, which reduces the level of personalisation that can be offered to the users.

The external approaches use an external adaptation engine, which provides the adaptation actions. In this approach, the self-adaptive software system consists of an adaptation engine and adaptable software. The external engine implements adaptation logic, mostly with the aid of middleware [14], [15], a policy engine [16], or other application-independent mechanisms. This provides a dedicated software component for performing the personalisation actions separated from the map application. Whenever the context model is changed, the personalisation engine can update the styling rules without performing any changes to the original code of the map application. To this aim, this article proposes the design of a self-adaptive map personalisation service from a self-adaptive map application and personalisation engine. This approach differs from the majority of existing works, which seek to embed the styling rules on the functional implementation of the map application. Moreover, integrating map personalisation with context-driven adaptation provides the map application with several benefits in terms of adaptability and dependability and increases their functionality to the end users. This article presents a self-adaptive map personalisation service developed using Context Oriented Software Development (COSD) [17]. COSD provides a generic development approach for building self-adaptive software from context-oriented components. COSD implements the principles of Model Driven Architecture (MDA) [18] and uses an adaptive middleware technology for supporting software with adaptability and dependability. The MDA approach provides a mechanism for designing software systems using an abstract model and facilitates software development by means of code generation. The adaptive middleware software component executes the adaptation actions and provides dynamic composition of context-oriented components based on the context information and user's profile. In the following section we discuss the concept of context-driven personalisation, which combines the personalisation process with dynamic context-driven adaptation.

III. CONTEXT-DRIVEN MAP PERSONALISATION

The software architecture of self-adaptive map personalisation is shown in Figure 1. The software integrates the map application with a Web Mapping Service (WMS)

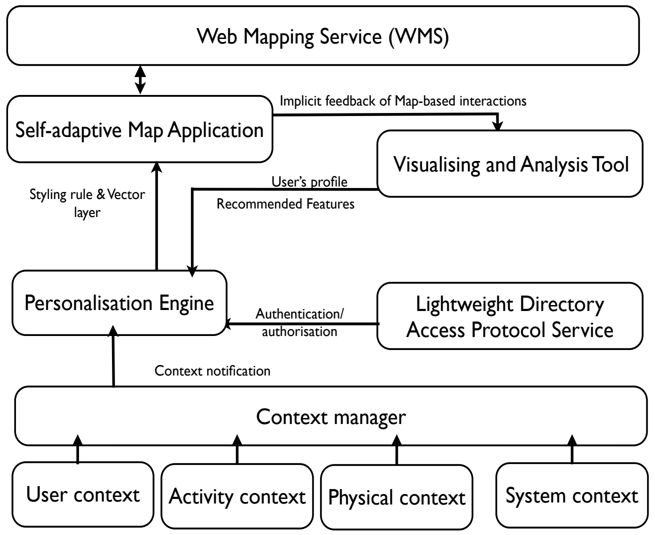
Web Mapping Service (WMS)
Self-adaptive Map Application
Implicit feedback of Map-based interactions
Visualising and Analysis Tool
Styling rule & Vector layer
User's profile
Recommended Features
Personalisation Engine
Authentication/ authorisation
Lightweight Directory Access Protocol Service
Context notification
Context manager
User context
Activity context
Physical context
System context

Fig. 1. Self-adaptive Map Personalisation Software Architecture

and personalisation engine. The personalisation engine monitors and interprets the users interactions with the map application. The web map usage is identified by a visualisation and analysis tool proposed in [1]. This service produces a dataset of geospatial features associated with a numerical weight (interest score) as calculated by the visualisation tool. The score attribute refers to a level of user interest on a specific spatial feature. Each feature is given value of zero for ”no interest”, 0.2 for ”low interest” , 0.5 for ”Avg Interest” and 1 for ”high interest” on its score attribute. This produces a set of recommended features for each user [5]. The self-adaptive map application integrates with the Lightweight Directory Access Protocol Service (LDAP), used to authenticate and authorise the user to use the system and obtain his profile data.

The profile includes personal information and interests, in the form of a set of features with interest score. When the personalisation engine receives the user's profile and the set of recommended features, it selects the best rule that suits the user's profile. The personalisation rule specifies a suitable map style and recommended features and passes them to the

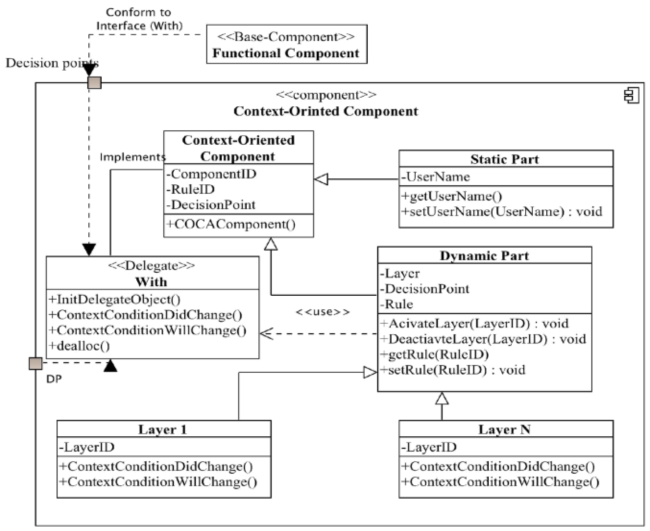
Conform to Interface (With)
<<Base-Component>>
Functional Component
Decision points
<<component>>
Context-Orinted Component
Implements
Context-Oriented Component
-ComponentID
-RuleID
-DecisionPoint
+COCAComponent()
Static Part
-UserName
+getUserName()
+setUserName(UserName) : void
<<Delegate>>
With
+InitDelegateObject()
+ContextConditionDidChange()
+ContextConditionWillChange()
+dealloc()
<<use>>
Dynamic Part
-Layer
-DecisionPoint
-Rule
+AcivateLayer(LayerID) : void
+DeactiavteLayer(LayerID) : void
+getRule(RuleID)
+setRule(RuleID) : void
DP
Layer 1
-LayerID
+ContextConditionDidChange()
+ContextConditionWillChange()
Layer N
-LayerID
+ContextConditionDidChange()
+ContextConditionWillChange()

Fig. 2. Context-Oriented Component Model

map application. At this stage, the map application uses the style to display the recommended features to the user.

Our map application architecture incorporates context-driven adaptation. According to the COSD paradigm, context-driven adaptation is achieved by designing the application from a set of components that follows the context-oriented component model [17]. Figure 2 shows a conceptual diagram of context-oriented component. A context-oriented component consists of three major parts: static and dynamic parts, and a delegate object. The static part implements a context-independent code fragment responsible for implementing the map core functionality. In other words, the static part implements any context-independent functionality, for example the map view and user login form are context-free functionality (i.e. context changes would not affect their functionality). The dynamic part implements the context-dependent functionality and participates in the context-driven adaptation. The context-dependent functionality refers to software functionality, which will exhibit volatile behaviour in the face of context changes. The dynamic part consists of multiple layers. Each component layer implements a specific context-dependent functionality, which implements a rule to be used for visualising specific map features. A layer is executed only if the associated context condition is found on the environment at runtime. For example, small display, low memory, and low battery are contextual conditions that need to be considered in the adaptation.

The personalisation engine invokes a specific rule in the execution that suits only the current context condition. For example, the personalisation service needs to reduce the amount of geospatial data to consider low battery condition on mobile devices. A rule is implemented inside a specific layer method. The method is associated with the low battery condition. This method will be executed whenever the personalisation engine receives the notification BatteryLevelWillChange, in this case, the personalisation engine executes the layer method, which implements a rule that displays less amount of geospatial data for the user (for example, displaying the features that have interest score between "0.7" and "1"). This approach can personalise the map content based on the scarcity of computational resources.

The context-oriented component is given an opportunity to do dynamic rule-based styling by executing different layers, which implement different rules. Each layer must implement two or more methods that encapsulate map rules and associate them with context conditions. For example, the two methods inside the layer class of the context-oriented component, could be ContextConditionDidChange and ContextCondition-WillChange. This allows the context-oriented component to perform personalisation actions about a specific context condition in active or proactive mode. Active personalisation refers to the ability of the personalisation engine to execute a rule when the condition is currently found in the execution context. Proactive personalisation refers to the ability of the personalisation engine to execute a rule that will handle a condition that can happen after a certain amount of time. An example of active personalisation is when the user's location is changed, the associated layer will execute a piece of code, that displays the user's location and his/her points-of-interest. On the other hand, an example of proactive personalisation is when the user's mobile is capturing high speed and acceleration. The personalisation engine can decrease the frequency of updates from the web mapping service until the speed of the mobile device is decreased to save allocated resources. More frequent updates of geospatial data is a power consuming process that needs more bandwidth and CPU throughput, consume the allocated resources and decrease the battery life.

The third part of the context-oriented component is the delegate object (see Figure 2). The idea of using a delegate object is that two components coordinate to solve a problem. A context-oriented component is general and intended for reuse in a wide variety of contextual situations. The base-component is the digital map object, which stores a reference to that context-oriented component (i.e. its delegate) and sends messages to inform the delegate (context-oriented component) that some context condition was changed. This gives the delegate an opportunity to de/activate a layer implementation (i.e. method implementation) dynamically [19].

In this sense, the map application is designed from a set of base-components (context-independent) and context-oriented components, which construct the personalisation engine. Whenever the map application notifies the personalisation engine about changes on user profile or context, the personalisation engine activates a layer implementation that adapts to the changes by specifying a styling rule that filters and prioritises geospatial data on the map. The following section describes a case study used to demonstrate different levels of personalisation and context-driven adaptations. This case study relates to the development of a self-adaptive map personalisation application in an eCampus environment.

IV. CASE STUDY: SELF-ADAPTIVE MAP APPLICATION

eCampus is a self-adaptive map application, which helps students, lecturers, staff members and visitors to explore the campus of the National University of Ireland, Maynooth (NUIM). In order for context-awareness capabilities to be made available, different aspects of spatial data need to be exploited including location, semantics, and time. The context-aware functionality offered in this application can be very useful to staff, students, visitors and the general public alike when navigating and otherwise interacting within our eCampus environment, specifically but also within any local environment generally while at home or on-site through web-based or smartphone connections respectively. There are two types of users: registered and non-registered users. A registered user includes anyone who wishes to log on (e.g. using their student/staff ID) and a non-registered user includes visitors to the campus (e.g. general public) who are not required to log on. As such, both user types are presented with different levels of functionality. The registered users profiles are recorded so that their personal timetables and interests can be displayed. This can be displayed with a grid/table of all activities for the user on the current day (course, lectures during the day/week) associated with the venue. The other option is that the schedule is directly displayed on the personalised map with overlay vector layer identify the geometry coordinates of the venue and detailed information about that activity. The recommendation of events is suited to each user profile (ie. based on their interests). The application selects the most relevant events to the user based on his profile. The features on the map refer to points-of-interest classified based on the score attribute (calculated based on the user's interest).

The development process and the design of eCampus including the self-adaptive map application and a personalisation engine are described in details in the following section. COSD is used to develop the eCampus self-adaptive application as shown in the following section, which demonstrate the eCampus ability to self-adapt its behaviour with respect to users' profile and context, and self-configure its functionality for styling the map content based on the available spatial data, allocated resources and quality of GIS services.

V. CONTEXT-ORIENTED SOFTWARE DEVELOPMENT (COSD)

In the development of the eCampus system, we employed COSD. COSD follows the principles of Model-Driven Architecture (MDA). In MDA, three different models are used during the three phases of software system design and development: the computation-independent model (CIM), the platform-independent model (PIM), and the platform-specific model (PSM). The CIM focuses on both the environment and the requirements of the system and hides the details of the software structure and processing. The PIM focuses on the operation of the system and hides details that are dependent on the deployment platform. The PSM combines the CIM and PIM, with an additional focus on the details of the use of a specific platform by the software system [18]. COSD focuses on partitioning the platform-independent model of the software into two views: the structure view (i.e. the base components) and the behaviour view (i.e. the context-oriented components). The structure view focuses on the base-structure of the map application and implements the context-independent functionality. The behaviour view focuses on modelling the context-dependent functionality of the software on individual layers of the context-oriented component. The design of a self-adaptive personalisation service for eCampus involves the following three phases:

Phase 1: Computation-Independent Model (CIM): A CIM is a model of a system that shows the system in the environment in which it will operate, and thus, it helps the developers to present exactly what the system is expected to do. It is useful not only as an aid to understand the software functionality, but also as a mechanism for predicting the exact behaviour of a software system as a result of runtime changes. The first step in the process is to understand the application's execution environment and provide a partial requirements diagram as shown in Figure 3.

In our case study, eCampus is required to adapt its behaviour and offer different levels of personalisation to the users de-pending on the available resources. For example, eCampus is required to adapt to the condition of low battery. This can be achieved by using a location service that consumes less power as shown in Figure 3. The "Battery level" requirement needs a rule to manage its context changes and display less geospatial data to suit the battery level. In other words, the map application monitors the battery level and the bandwidth connectivity. If the battery level is high and healthy, eCampus uses the GPS service with more accurate location's update, displays more features to the user, and animates the features' appearance on the map when they are recommended. If the battery level is less than a specific limit, a WIFI-based location is used to save the battery energy and obtain more geospatial data. Finally, if the battery level is low, the eCampus application switches off the GPS or WIFI location services and uses the cell-tower location services. Using a cell-tower location reduces the accuracy of the location but saves battery energy. In addition, the application may reduce the number of features it displays on the map based on different interest score levels (values).

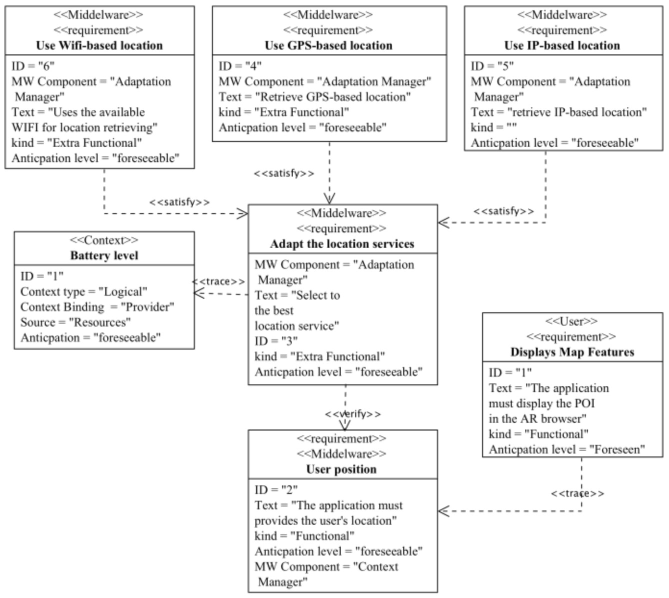
<<Middelware>>
<<requirement>>
Use Wifi-based location
ID = "6"
MW Component = "Adaptation Manager"
Text = "Uses the available WIFI for location retrieving"
kind = "Extra Functional"
Anticpation level = "foreseeable"
<<Middelware>>
<<requirement>>
Use GPS-based location
ID = "4"
MW Component = "Adaptation Manager"
Text = "Retrieve GPS-based location"
kind = "Extra Functional"
Anticpation level = "foreseeable"
<<Middelware>>
<<requirement>>
Use IP-based location
ID = "5"
MW Component = "Adaptation Manager"
Text = "retrieve IP-based location"
kind = ""
Anticpation level = "foreseeable"
<<satisfy>>
<<satisfy>>
<<satisfy>>
<<Context>>
Battery level
ID = "1"
Context type = "Logical"
Context Binding = "Provider"
Source = "Resources"
Anticpation = "foreseeable"
<<trace>>
<<Middelware>>
<<requirement>>
Adapt the location services
MW Component = "Adaptation Manager"
Text = "Select to the best location service"
ID = "3"
kind = "Extra Functional"
Anticpation level = "foreseeable"
<<User>>
<<requirement>>
Displays Map Features
ID = "1"
Text = "The application must display the POI in the AR browser"
kind = "Functional"
Anticpation level = "Foreseen"
<<verify>>
<<requirement>>
<<Middelware>>
User position
ID = "2"
Text = "The application must provides the user's location"
kind = "Functional"
Anticpation level = "foreseeable"
MW Component = "Context Manager"
<<trace>>

Fig. 3. Partial Requirements Diagram

Phase 2: Platform-independent model (PIM): The platform-independent model focuses on the operation of a system while hiding the details necessary for use of a particular platform. In this phase, the requirements diagram is combined into a use-case model. The use-cases describe the interactions between the software system and the actor. The system-dependent and environment-dependent behaviours are modelled as an extension of the functional use-cases, which refers to the structure view of the software. The functional use-cases are modelled in a class diagram describing the application core functions. The extended use-cases are modelled as another class diagram that describes the application's behavioural model, which refers to the behavioural view of the software.

The requirements diagram in Figure 3 represents the main inputs for this task. Each requirement is incorporated into a use-case, and the developers identify the actor of the requirement. An actor could be a user, system, or context condition. The use-cases are classified into two distinct classes, i.e., the core functionality and extended use-cases, by the context conditions. The first step is to identify the interaction between the actor and the software functions to satisfy the user requirement in a context-free fashion. For example, displaying the map view is context-independent in the sense that the application must provide it, regardless of the context conditions. All these use-cases are modelled separately, using a class diagram that describes the application core-structure or the base-component model. The class diagram is modelled independently from the variations in the context information.

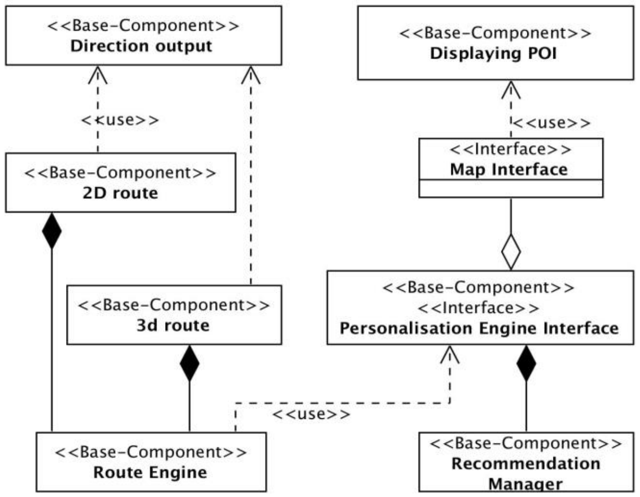

Fig. 4. Partial Requirements Diagram

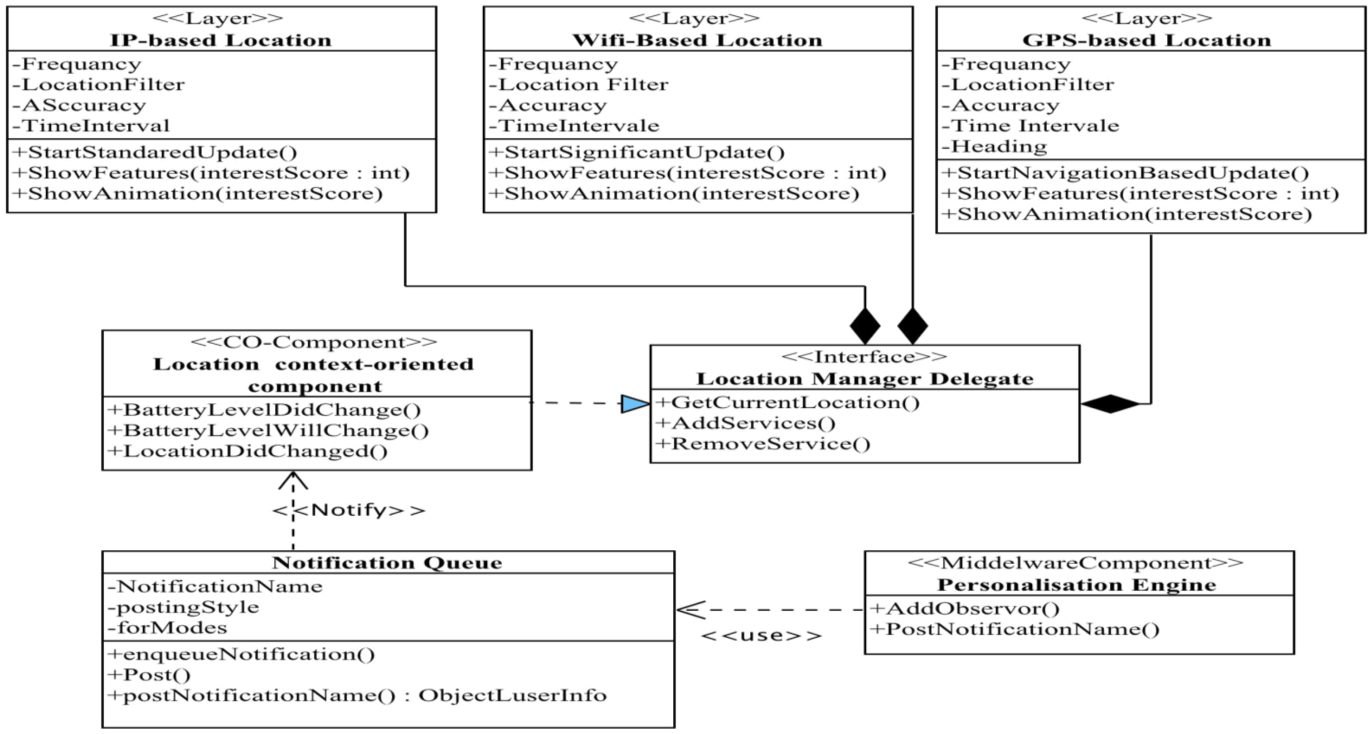

Fig. 5. Location context oriented component with sublayers

For the eCampus scenario, some classes, such as "Displaying POIs", "Route UI", "MapUI", and "User Interface", are classified to be on the application core (i.e. the structure view of the software). These classes provide the core functions for the eCampus user. Figure 4 shows the core-structure class-model without any interaction with the context environment. Figure 5 shows a context-oriented component modelled to anticipate the 'location service'. The context-oriented component implements a delegate object and sub layers; each layer implements a specific context-dependent function. Thersonalisation engine uses this delegate object to redirect the execution among the sub layers, based on the context condition and the interest score for each feature obtained from the user's profile. For example, the styling rule shown in Listing 1 implements a personalisation action to handle the low battery condition. If the battery level is high, the personalisation engine executes StartNavigationBasedUpdate() method which uses GPS-based location service. At the same time, it will execute ShowFeatures() method, which displays all features that have an interest score between 0.3 and 1. In addition, it will animate the map features using ShowAnimation() method, a

Phase 3: Platform-specific model (PSM): The three diagrams modelled in the previous tasks are transformed into Platform Specific Model (PSM). PSM focuses on the configurations of the target platform. The specifications for mobile devices are different from desktop platforms. At this stage, each platform has a separated implementation of the eCampus application. The Software developers integrate the target platform configurations with the following models: 1) The behavioural model, which includes the context-oriented object diagram of eCampus 2) The core structure of the eCampus application. 3) The context meta-model, which includes the context model itself and the context and user's profile models, that describes the available context entities and their representation on the target platform. Using a code generation tool the final code of eCampus is ready to be deployed on the target platform.

VI. ECAMPUS IMPLEMENTATION AND EVALUATION

This section focuses on evaluating the performance of the eCampus application, which was implemented using COSD and the personalisation engine. To this aim, the eCampus application was implemented in two different versions. The first version was implemented using the COSD personalisation engine. This version is called COSD-eCampus. The second version implemented using static styling rules embedded in the application code for performing static-based personalisation. This version is called eCampus in the experiment. For the purposes of the evaluation, both applications COSD-eCampus and eCampus were executed on an IPhone device [20]. During the experiment, the CPU activity, CPU time, and energy usage were measured for evaluating the cost of the personalisation process. These values were measured using the energy diagnostics and activity-monitoring tools, which analyses the CPU usage of the running application on the IPhone device, and generates cost and performance report. The Energy Diagnostic tool was used to measure the battery while the device was not connected to an external power supply; after the experiment was finished, the data were imported from the iPhone and then analysed using the activity-monitoring tool. The battery life has been measured by running each version of the eCampus application on IPhone device for 12 hours.

```
    if (BatteryLevel >= 50) then {
         StartNavigationBasedUpdate() &
              ShowFeatures( interestScore between
              0.3 and 1} & ShowAnimation(
              interestScore > 0.2)
          &  displayRecommendation(Events)}
      else if (BatteryLevel <=50 & BatteryLevel
         >=20)
      then {StartSignificantUpdate() & ShowFeatures(
         interestScore >= 0.5) & ShowAnimation(
         interestScore > 0.5) &
         displayRecommendation(Events)}
      else if( BatteryLevel < 20)
      then {StartStandaredUpdate() & ShowFeatures(
         interestScore  >= 0.7 )};
```

Listing 1. Styling Rule

For example, at runtime, the context-oriented component Location (see Figure 5) registers itself with the context manager to be notified when the BatteryLevelDidChange, CPULevelDidChange, MemoryLevelDidChange, DeviceOrientationDidChange and/or LightLevelDidChange. When the manager posts the notification [BatteryLevelDidChange] to the "Location" component, the personalisation engine reads the styling rule (see list 1.1). Based on the styling rule, the personalisation engine calls the delegate object (Location Manager Delegate), which forwards the method invocation to the chosen sublayer, and it executes the "ShowFeatures()" method, which displays a specific subset of the map features. For example, if the battery level is less than 20%, then the personalisation engine activates the sublayer "IP-based location" and executes the method "ShowFeatures()", which displays all features that have an interest score between 0.7 and 1.

The Energy usage experiment shows that the COSD-eCampus application saves the battery consumption by 28% for the personalisation process, despite its ability of self-adaptability and dynamic rule-based styling, as shown in Figure 6. One of the expected benefits of using COSD in developing the self-adaptive application is the enhancement of the personalisation process, context monitoring and detection processes. The eCampus implementation using static-based personalisation consumes more energy during the personal-isation process, thus draining the battery faster, because a large amount of map features are rendered to the map, which needs more CPU time, memory allocation and more energy. On the other hand, when the COSD-eCampu s performs the personalisation process, the application is able to personalise the map content and enhance the amount of map features based on the context condition. The personalisation/adaptation time and the context handling time are shown in Figure 7. The personalisation time in COSD-eCampus was less than eCampus implementation, because the static-based personal-isation process needs more time to process large volumes of map features and to render them on the map. In the COSD-eCampus, the application selects specific features and processes them based on the availability of resources. For this reason, the personalisation time was reduced by 38 milliseconds in the COSD-eCampus implementation, which improves the personalisation process by 57%. In addition, handling the context events was less in the COSD-eCampus as it implements a dedicated context manager for processing the context events. In the eCampus implementation, there was no specific manager implemented for handling the context events, which forces the eCampus application to process large amounts of context events and do the personalisation at the same time.

VII. CONCLUSIONS AND FUTURE WORK

This article describes a dynamic rule-based personalisation engine implemented at the client side of the architecture. It enables map applications to filter and prioritise geospatial data using dynamic decision-making processes, which consider users' profiles and context for selecting and styling map features according to their needs. Using an external engine for personalising map content provides map applications with several benefits. The personalisation engine can update the rules' syntax based on the evolution of the user's profile and his interest without changing the map application's code. Separating the rules' code from the application code requires less intervention from the software developers and reduces the maintainability effort. The map application offers different levels of personalisation for different users, as it can handle a wide range of users' profiles without encoding the rules on the map application's code. Sending specific personalisation requests to the Web Mapping Service (WMS) increases the efficiency of personalisation. The map application can request specific map features from the WMS, which uses less resource.

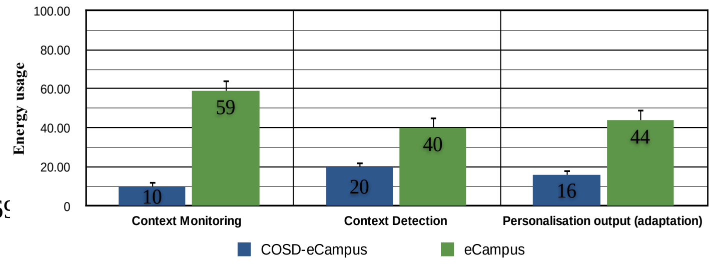

Fig. 6 Energy usage for eCampus application

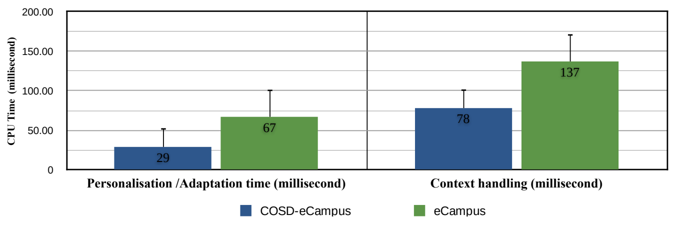

Fig. 7. Personalisation time (ms)

Our current prototype of the rule-based engine requires improvement on several aspects including rules mismatch and resolution. This is in line with providing runtime verification and evaluation mechanism, which can verify the validity of personalisation output among the users tasks, requirements and needs. In addition, there is a need to measure the user's experiences and satisfaction after using the personalisation services, which can improve the quality of map personalisation

for the future. These improvements form part of our future work.

VIII. ACKNOWLEDGEMENT

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the National Development Plan. The authors gratefully acknowledge this support.

REFERENCES

[1] A. Tahir, G. McArdle, and M. Bertolotto, "Visualising User Interaction History to Identify Web Map Usage Patterns," in Proceedings of 14th AGILE International Conference on Geographic Information Science, Utrecht, Netherlands, April 2011, pp. 46 – 52.

[2] D. Wilson, M. Bertolotto, and J. Weakliam, "Personalizing map content to improve task completion efficiency," Int. J. Geogr. Inf. Sci., vol. 24, no. 5, pp. 741–760, May 2010. [Online]. Available: http://dx.doi.org/10.1080/13658810903074490

[3] M. Hirose, R. Hiramoto, and K. Sumiya, "Geminimap - geographical enhanced map interface for navigation on the internet," in Web and Wire-less Geographical Information Systems, ser. Lecture Notes in Computer Science, J. Ware and G. Taylor, Eds. Springer Berlin Heidelberg, 2007, vol. 4857, pp. 279–292.

[4] G. McArdle, A. Ballatore, A. Tahir, and M. Bertolotto, "An open-source web architecture for adaptive location based services," In Proceedings of the 14th International Symposium on Spatial Data Handling (SDH), Hong Kong, vol. 38, no. 2, pp. 296–301, 2010.

[5] A. Ballatore, G. McArdle, C. Kelly, and M. Bertolotto, "RecoMap: an interactive and adaptive map-based recommender," Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 887–891, 2010.

[6] M. Albanese, A. Picariello, C. Sansone, and L. Sansone, "Web personalization based on static information and dynamic user behavior," in Proceedings of the 6th annual ACM international workshop on Web information and data management, ser. WIDM '04. New York, NY, USA: ACM, 2004, pp. 80–87. [Online]. Available: http://doi.acm.org/10.1145/1031453.1031469

[7] S. E. Middleton, N. R. Shadbolt, and D. C. De Roure, "Ontological user profiling in recommender systems," ACM Transaction Information Systems, vol. 22, no. 1, pp. 54–88, Jan. 2004. [Online]. Available: http://doi.acm.org/10.1145/963770.963773

[8] R. Hirschfeld, P. Costanza, and O. Nierstrasz, "Context-oriented pro-gramming," Journal of Object Technology, vol. 7, no. 3, pp. 125–151, March 2008. P. Oreizy, M. Gorlick, R. Taylor, D. Heimhigner, G. Johnson, N. Med-vidovic, A. Quilici, D. Rosenblum, and A. Wolf, "An architecture-based approach to self-adaptive software," Intelligent Systems and Their Applications, vol. 14, no. 3, pp. 54–62, 1999.

[9] Z. Yu, Y. Wang, and L. Fang, "Study on the intelligent map service for adaptive geo-visualization," in the Proceedings of the 18th International Conference on Geoinformatics, ser. (Geoinformatics, 2010), Beijing, China, june 2010, pp. 1 –6.

[10] A. Emrich, A. Chapko, and D. Werth, "Context-aware recommendations on mobile services: the m:ciudad approach," in Proceedings of the 4th European conference on Smart sensing and context, ser. EuroSSC'09, Guildford, UK, 2009, pp. 107–120.

[12] M. Claypool, P. Le, M. Wased, and D. Brown, "Implicit interest indicators," in Proceedings of the 6th international conference on Intelligent user interfaces, ser. IUI '01. New York, NY, USA: ACM, 2001, pp. 33–40. [Online]. Available: http://doi.acm.org/10.1145/359784.359836

[13] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, and E. Gjorven, "Using architecture models for runtime adaptability," IEEE software, vol. 23, no. 2, pp. 62–70, 2006.

[14] T. Chusho, H. Ishigure, N. Konda, and T. Iwata, "Component-based application development on architecture of a model, ui and components," in Proceedings of the Seventh Asia-Pacific Software Engineering Con-ference Conference, ser. (APSEC '00), Singapore, 2000, pp. 349–358.

[15] A. Mukhija and M. Glinz, "The casa approach to autonomic applica-tions," in Proceedings of the 5th IEEE Workshop on Applications and Services in Wireless Networks, ser. (ASWN 2005), Paris, France, June–July 2005, pp. 173–182.

[16] R. Anthony, D. Chen, M. Pelc, M. Perssonn, and M. T. rngren, "Context-aware adaptation in dyscas," Electronic Communications of the EASST, vol. 19, p. 15, 2009.

[17] B. Magableh and S. Barrett, "Context oriented software development," Journal of Emerging Technologies in Web Intelligence (JETWI), vol. 3, no. 4, pp. 206–216, June 2011.

[18] A. G. Kleppe, J. Warmer, and W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise. Boston, MA, USA: Addison-Wesley Longman Publishing, 2003.

[19] E. Buck and D. Yacktman, Cocoa design patterns, 2nd ed. Developer's Library, 2010.

[20] "Ios 4.0 apple developer library," http://developer.apple.com/library/ios/navigation/, 2011, "[Online; accessed 1-April-2011]".