Using Intentions and Plans of Mobile Activities to Guide Geospatial Web Service Composition

Bo Yu and Guoray Cai†
Spatial Information and Intelligence Laboratory, College of Information Sciences and Technology
Penn State University, University Park, PA 16802, USA
Email: {byu, cai}@ist.psu.edu

Abstract -- Mobile applications are increasingly taking advantages of the diverse geospatial web services to meet the information needs of their users. However, matching available web services to user's information needs is not a trivial task, as there are many contextual factors that may influence the fitness of use. In addition, mobile activities can be highly dynamic and interleaving, which demand certain level of context-adaptation for web service matching policies. Previous work on context-based service matching and composition tends to focus on environmental and functional contexts that can be either sensed directly or defined a prior, and they assume relatively stable user activities. In this paper, we describe a method for representing and reasoning the intentional structures of mobile activities and use it to contextualize mobile map services. Our context model treats a user’s mobile activity as an evolving collaborative plan situated in a set of physical and mental factors. The model explicitly reasons on the intentional structure of the mobile activity to determine the appropriate service matching policy on the fly. The feasibility and benefit of using this model is demonstrated through the implementation of a prototype system, MyTour - a mobile city tour guide application.

Keywords—geographical information; web service composition; activity; context adaptation.

I. INTRODUCTION

Geographical web services are modular web applications that provide services on geographical data, information, and knowledge [38]. With the availability of geographical web services and associated architectures [32] and standards [30], developing advanced geospatial applications becomes a process of orchestrating (or composing) heterogeneous web services for supporting spatial decision-making, knowledge discovery, and spatial activities [11, 35]. Approaches for automatic or semi-automatic composition of atomic services to generate value-adding services have been proposed in the past, ranging from pure feature matching, syntactic chaining, to more sophisticated semantic-based methods. While business communities took a more workflow-oriented approach [23], semantic web communities recently followed ontology-based [20] and AI planning based approaches [31]. These approaches are far from offering a comprehensive solution to the overall challenges of automated composition of web services into a coherent solution. The difficulties in representing and reasoning on semantics and domain processes remain to be addressed.

Mobile map services that provide relevant geographic information to people on the move represent one of the fundamental and most widespread types of location-based services (LBS) [28]. The problem of composing geo web services for mobile applications is unique for two reasons. First, compared with other types of data, geographical information is more heterogeneous in its media, representational forms, and semantics[4, 22]. There is little agreement on the ontology of geographical objects and processes, except at the most abstract level [1]. Second, mobile applications must have a high degree of adaptability of its behavior to the changing environment and work contexts [21]. This makes it difficult to use traditional web service composition methods, since they all assume prescribed actions, states, goals and events. On the other hand, mobile geographical applications offer a number of advantages and opportunities over other (desktop) applications [8]. In particular, human mobile activities are highly driven by their goals which can better be sensed or reasoned from the intentional structure of the larger activity. Such knowledge of mobile activity offers rich and detailed contextual clues that can be used by a service composition engine to make real time inference on the information needs. The key to the success is to build the awareness the ongoing activity into the web service composition engine so that web service requests can be flexible and adaptive.

In this paper, we present our idea of contextualizing mobile map service composition in terms of activities. User activities become first-class entities that are represented explicitly by the mobile map services composition engine. Following the cognitive and mental state views of tool-mediated human activity [33, 36], our model of activities goes beyond traditional AI planning models to include intentions and belief towards interrelated goals as more stable and relevant contexts. This model of activity is available to the service composition engine and allows the engine to reason about what and when geographical information services are needed and how it influences the behavior of the user and the system.

To motivate subsequent discussions, we start with a scenario where a traveler plans a city-tour with the assistance of map-based tour-guide that provides geographical information services (see Table 1). From this scenario, we can differentiate four roles that geographical

† Corresponding author
information serves: (1) as part of the spatial constraint on the user’s mobile activity, (2) as the criterion for promoting visual saliency to support the user’s activity, (3) as knowledge-precondition for action, or (4) as indication of activity state changes.

Table 1 describes the scenario as four episodes, each of which was analyzed to understand the information needs. The selection and composition of geo web services is determined by the information need to advance user’s activity from current state to future states. As the users make progress on their activities, their intentional focus shifts and new geo information service must be composed correspondingly. It is the activities supported by mobile map services that play a central role in deciding when and why user’s location becomes an important factor to influence the system’s supporting behavior, and how the location information is consumed to provide the support.

### User Activity Development

| Spatial constraint. The system can use Jim’s current location together with the time constraint (half a day) to estimate the maximum distance that Jim could go and use this as a spatial constraint to retrieve POIs and calculate the appropriate map extent. |
| Knowledge for action. Jim needs to have knowledge about the route from the current location to the destination. As a result, the system can highlight the current location on the map to help him finish his task. |
| Representing activity state. Jim needs to monitor where he is at in relation to the destination. The map service update itself to show the status of the navigation task. |

The following sections of this paper present our approach in detail. After a brief review of related work (section II), we present our computational model to represent user activities based on the SharedPlan theory (section III). In section IV, we demonstrate how this model can be employed in the motivating scenario to infer different roles of location information and how it influences the map-based support. Section V provides our prototype system MyTour design and implementation details. We conclude the paper by discussing the advantages and limitations of our approach, as well as possible future work.

II. RELATED WORK

A. Geo Web Service Composition

Composing web services for an application can be analyzed as a five-phase process, according to Kim et al. [19]. In the first phase (specification phase), users specify their intended goal of their domain activity to be supported by the composed web services, and produce an abstract specification. This involves the specifications of goals, tasks, events, states, and constraints as models of the domain. Then, the second phase (planning phase) takes abstract specification of the domain activity and generates a formal representation of process flows that allow machines to automate service composition. Major approaches in this phase include workflow-based composition, template-based composition, and AI planning techniques (such as Markov chains, backward chaining, and graph theory-based techniques for service chaining. The third phase conducts syntactic and semantic validation of the service composition workflow to ensure the composed services satisfy the stated goals of the users. The last two phases are service discovery and execution, which will not be the focus of this paper.

In real applications, the first three phases are commonly done by human developers. Such work is time-consuming and error-prone, but most importantly, it requires developers to have good knowledge in both the intended application domain (e.g. wildlife protection) and the technical domain (e.g. geographical information services). Recently, automate service composition methods with the use of AI planning techniques have emerged as promising research directions [31, 35]. However, computational work mostly deals with problems within one of the phases, and they rarely address cross-phase issues. This is mostly due to the difficulties in bridging the huge semantic gap going from the specification phase to the planning phase. The exceptions are recent work in context-based methods for matching and discovering web services [27, 34]. Context is potentially an idea that relates the semantic issues across different phases. Unfortunately, the work on context is limited to some broad categorizations and representation languages, with little influence on service composition.

B. Context and Activity

Contexts affect all phases of service composition. However, the intuitive concept of ‘context’ is quite tricky to define and formalize. A commonly adopted definition of context, as offered by Dey [9] is “all things that are relevant to the interaction between a user and an application.” This definition is too broad to be useful as a formal definition. Other researchers have focused on a subset of all possible contextual factors that have predictive power over an
application’s behaviors [7]. It is important to distinguish between context and situation. While a "situation" is an observer-independent and potentially unlimited resource that characterize an actual setting within which a tool is used, a "context", as an expression of certain interpretation of a situation, is observer-dependent and includes a relatively small subset of features that has impact on the behavior of the system [26]. A key point is that relevant features of a context may be highly application-specific. This is consistent with the interactional view of context [10], which argues that something is part of the context because it is used for adapting the interaction between human and the current system, and features of the world become context parameters through their use [37]. Hence, the most important context of information services is the activity within which information is used.

The idea of organizing geographical information support in terms of activities has been discussed in some location-based systems. Liao et al [24] uses location data from a wearable GPS location sensor to identify a user’s significant places, and learns to estimating high-level activity categories, e.g. working, shopping, and dining out. Such activity knowledge can then be used to adapt the system’s behaviors. However, the concept of activity used in these systems is lightweight in the sense that they only consider an activity as consisting of a range of actions performed with a collection of computational support [3]. Huang and Gartner [15] provided an example about how a model of human activity can be used to identify relevant context parameters in designing context-aware pedestrian way-finding services. However, all of these existing efforts made only limited advances in modeling and using activities for automated reasoning - they either discussed activity at the conceptual level or applied it at the design stage. None of them have explored the potential of modeling the human activity in a computational way so that the system can maintain an updated activity model and reason about the relevant contextual factors on the fly. Our work attempts to develop a computational model of user activities that can be used by service composition engine to computationally determine when and how geographical information is needed during an ongoing activity.

### III. MODELING SITUATED ACTIVITIES

In order to model an activity, we first need to understand what an activity entails. In our approach, we subscribe to Activity Theory to conceptualize human activity [29]. From this perspective, an activity is a dynamic construct expressed over a period of time, and it consists of people, artifacts, an object or motive, socio-cultural rules, and roles. An activity usually starts with an objective as motivations behind the activity. Multiple actions are performed to reach the overall objective. Each action is driven by a conscious intentional goal. Finally, actions are performed by operations that are unconscious, often routine actions carried out automatically as a service. The conceptualization of activity from this perspective provides several requirements for the computational model of activity [18]:

1. Activities are rooted in mental attitudes of individual participants. It is the goals and intentions of participants that lead them to plan and perform intended actions. As a result, to model a human activity, a key is to characterize the mental attitudes of participants in an activity.
2. Activities are hierarchically structured from motive-driven high-level activities through goal-driven actions down to operations without being goal-directed. Each of these three levels can itself consist of several layers of complexity. Hence, the computation model of an activity must be able to capture the hierarchy of subsidiary actions and operations in an activity.
3. Activities are highly dynamic and time-dependent. As users continue their activities, the states of activities keep changing. Therefore, the model of activity must be able to update itself accordingly.

To satisfy these requirements, we model an activity as a evolving collaborative plan between the user and the system, following the computational theory of SharedPlans [14]. In particular, (1) we model the mental state of an activity as the intentions, beliefs, and mutual beliefs of the participants; (2) we model an activity as hierarchically structured plans that are dynamically constructed and executed under specific situational and resource constraints; (3) we model the development of an activity as the evolution from a partial shared plan (PSP) to a full shared plan (FSP) through elaboration.

In the rest of this section, we first describe how a activity can be formally represented as a shared plan, followed by a description of the development cycle of a shared plan capturing the dynamics of an activity.

#### A. The Representation of Activities

In our approach, we represent an activity as a shared plan. In general, a shared plan is a moment-to-moment representation of an unfolding activity. It includes hierarchically organized plans and subplans that form a PlanGraph (see Figure 1).

![Graphical representation of PlanGraph](image)

To model the necessary mental attitudes, each node in a PlanGraph includes several slots to store the system’s beliefs about other agents’ mental states: (a) Intentions are slots recording the system’s beliefs about intentions of each agent towards the associated action; (b) Capability indicates the system’s belief about the ability to perform the action,
such as whether they can identify a recipe or they can bring it about; (c) Beliefs are slots for recording the system’s beliefs about what the other agents also believe about this action. Reasoning on the changes of mental attitudes is performed through a set of mental state operators as specified in the SharedPlan theory [14]. Unlike traditional notion of AI plans that were abstract expression of human actions, our notion of plans refers to situated actions. Specifically, we consider that each situated action has both situation-independent and situation-dependent aspects. For example, although there is a generic set of knowledge about how to cook a dinner (assuming one knows a set of recipes), exactly what to cook depends on the available resources (meat, vegetables, ingredients, etc) as well as the skill and time of the person. This is known as the “knowledge precondition principle” of collaborative plans [25]. In a PlanGraph, we handle the knowledge-conditions as a special type of node - parameters. Nodes with oval shape in Figure 1 indicate parameters, and nodes with rectangle shape represent subsidiary actions. A plan underneath a parameter node is the plan for identifying that parameter.

The theory of situated actions and plans [2, 36] offers insights on how information services (as part of the situational context) condition or constrain human activities.

B. Representing the Developmental Aspects of Activities

An activity normally starts with an overall purpose or objective that agents intend to accomplish. This is often represented as the root node in a PlanGraph. As the activity proceeds, the human user and the system collaborate (communicate and interact with each other) toward the success of their shared goal in the activity [6]. The development of the activity is the process that iterates four steps of reasoning (see Figure 2): (1) recognition; (2) explanation; (3) elaboration; and (4) behavior generation.

In the plan explanation phase, the system analyzes each input to infer its beliefs about the changes in the user’s mental states, the activity, or environment. Then the system attempts to explain how the meanings of these new beliefs relate to the current PlanGraph. If the new beliefs can be successfully explained, the system updates the PlanGraph to accommodate these new beliefs. In the plan elaboration phase, the system needs to adopt or update its mental states according to these changes, do the means-end reasoning to elaborate the shared plan, and perform individual actions to advance the activity.

1) Recognition. This step refers to the process that the system recognizes the input from the users or the environment and establishes corresponding beliefs towards them. The inputs can come from the user’s explicit requests, such as selecting a menu item from the client or a speech utterance in a spoken dialogue system; or are collected implicitly by the system through different sensors, e.g. the GPS sensor that detects the location changes of the user.

The system searches the domain knowledge base for an appropriate match between the inputs and the meanings through three levels of interpretation [16] [13]: lexical, syntactic, and semantic. The result of recognition is a set of new beliefs \( P \), indicating how the system understands the new inputs.

2) Explanation. The second step is to determine how these new recognized beliefs contribute to the collaborative activity as augmenting the partial shared plan. This process starts from the root node of the PlanGraph and traverse the nodes to decide whether the new beliefs can contribute to any node of the PlanGraph. For each node traversed, the system must decide if the new belief from the input contributes to the current plan node. To model this step, we introduce two subsidiary processes to determine the relations between a belief \( p \) and a plan node. A-Contributes determines the relationship between \( p \) and an action, and P-Contributes determines the relationship between \( p \) and a parameter.

![Figure 2. Reasoning process with PlanGraph](image1)

**Explanation** (Proposition \( \rho, \text{PlanNode } \alpha \))

1. if \( \alpha \) is an action, and \( \rho \) contributes to \( \alpha \) in certain way, i.e. A-Contribute(\( \rho, \alpha \)) is true, current action \( \alpha \) explains \( \rho \);
2. if \( \alpha \) is a parameter, and \( \rho \) contributes to \( \alpha \) in certain way, i.e. P-Contribute(\( \rho, \alpha \)) is true, parameter \( \alpha \) explains \( \rho \);
3. otherwise, \( \rho \) does not contribute directly to current action \( \alpha \). For each subsidiary parameter and action of \( \alpha \) (\( \beta_1, \beta_2, ..., \beta_k \)), repeat the augment process: explanation (\( \rho, \beta_1 \)).

Where: A-Contribute(Proposition \( \rho, \text{PlanNode } \alpha \)) could mean four things:
1) if \( \rho \) indicates the initiation of a subsidiary shared plan for action \( \alpha \), and return true;
2) if \( \rho \) indicates the completion of the current shared plan for \( \alpha \), the system will ascribe the following belief to \( \alpha \):
   \[\text{Bel(Sys,Bel(User, FSPG(\alpha, \rho)), t))}\]
   and return true;
3) if \( \rho \) indicates that \( \delta \) is a sub-action of current action \( \alpha \), the system will ascribe the following belief:
   \[\text{Bel(Sys,Bel(\text{User } \delta \in \text{Recipe}(\alpha)))}\]
   and return true;
4. if \( \rho \) represents a piece of information that is part of the performance context of current action \( \alpha \), the system will ascribe the following belief:
   \[\text{Bel(Sys,Bel(\text{User } \delta \in \text{Val}(\alpha)))}\]
   and return true;

P-Contribute(Proposition \( \rho, \text{ParamNode } \alpha \)) could mean one of the following:
1. if \( \rho \) refers to a value \( v \) of the current parameter \( \alpha \), the system will ascribe the following belief to \( \alpha \):
   \[\text{Bel(Sys,Bel(\text{User } \delta \in \text{Val}(\alpha)))}\]
   and return true;
2. if \( \rho \) indicates that \( \alpha \) is a sub-action of identifying the current parameter \( \alpha \), the system will ascribe the following belief:
   \[\text{Bel(Sys,Bel(\text{User } \delta \in \text{Recipe}(\text{id.param}(\alpha)))}\]
   and return true;

![Figure 3. Plan explanation algorithm](image2)
further updated to accommodate these changes, because the
new beliefs might imply chained changes to other related
actions.
3) Plan Elaboration. After the plan explanation process, the
context of the activity is changed. Therefore, the system
needs to elaborate the PlanGraph to accommodate the
changes and advance the collaborative activity from the
system side. The elaboration process begins with the root
node of the PlanGraph and adopts the depth-first traverse
to visit the whole plan based on these reasoning rules. The
elaboration ends when no more parts of the PlanGraph can
be further elaborated. Reasoning rules for plan elaboration
follow the principles of shared cooperative activities [5] and
include:

(1) Recipe Selection: the system intends that the group
d will develop a FSP to select a recipe for the action:
\[
\text{IntTh}(\text{Sys}, FSP(\text{GR}, t_{day}, C_{al}))
\]
\[
\Rightarrow \text{IntTh}(\text{Sys}, FSP(\text{GR}, \text{select recipe}(t_{select}), C_{select}))
\]
(2) Constraint Satisfaction: it requires that all the
members of the group be committed to making sure that the
constraints for doing $\alpha$ will hold.
\[
\text{IntTh}(\text{Sys}, FSP(\text{GR}, t_{day}, C_{al})) \Rightarrow \text{IntTh}(\text{Sys}, \text{const}(t_{day}))
\]
4) Behavior generation. Following the cooperative nature of
SharedPlans [17], the system needs to adopt corresponding
commitments to exhibit helpful behaviors. In general, two kinds of helpful behaviors can be provided by the
system: performing domain actions that are helpful to
the user’s task, and communicating relevant information to
ensure the user’s success in doing the action [17].

IV. ACTIVITY-MEDIATED ADAPTATION OF WEB SERVICE
COMPOSITION

By modeling the user’s activity in a PlanGraph as
described in Section III, we believe that web service
composition can be designed to adapt to the changing
activity state. In this section, we particularly focus on the
geographical information needs of advancing activity and
demonstrate how maps play different roles in the
development of the shared plan of an activity.

We now return to the motivating scenario in Section I.
Figure 4 depicts the higher-level view of the PlanGraph
(mental attitudes slots associated with each node are omitted
for simplicity), which indicates the major goals of the user
in the development of the activity.

Figure 4. Upper-level part of the PlanGraph in the scenario
[Note: Nodes are colored ‘blue’, ‘red’, or ‘mixed’ to represent the
involvement of ‘system only,’ ‘user only,’ or ‘both user and system,’
respectively]

This PlanGraph reveals that the user’s top-level goal to
tour a place is decomposed into three major goals: (a) identify
the place to go, (b) plan the trip, and (c) navigate to
the destination. Each of these goals can be further divided
into subsidiary actions that may be performed by the system
(in blue), the user (in red), or the group together (in both
colors). During the different development phases of this
PlanGraph, we can identify points in time when geographical
information services are needed.

Next, we will analyze the four episodes of the scenario
to illustrate the change of system’s behavior. The first episode
(in Table 1) happens when the user indicates that the overall
goal is to take a tour with the time constraint:
\[
\text{Bel}(\text{Sys}, \text{IntTh}(\text{User}, FSP(\text{GR}, \text{take_tour}, t_{tour}, C_{tour})))
\]
\[
\text{Bel}(\text{Sys}, \text{duration = half a day} \in \text{const restr(take_tour)})
\]
From these beliefs, the system commits itself to the
group activity:
\[
\text{IntTh}(\text{Sys}, FSP(\text{GR}, \text{take_tour}, t_{hour}, C_{take_tour}))
\]
This commitment leads the system to adopt further
intentions based on the elaboration rules described in
Section III. The result of such reasoning is shown as the
PlanGraph in Figure 5.

Figure 5. The PlanGraph at episode I of the scenario

The system further elaborates the plan by adopting a
‘recipe’ (i.e. a strategy to accomplish a goal) which can be
found from the knowledge base. When it comes to the
behavior generation phase, search possible places for the
purpose of helping the user to identify the place to go.
During the elaboration process of the complex action
“Search nearby places”, the temporal constraint of the
overall activity (‘half-day’), which was given earlier, is
explained into the plan for calculation of the distance extent
of the tour. Figure 6 shows the state of the activity plan up
to the moment when the destination of the tour (Stadium) is
determined (end of the second episode). Based on this plan,
our service composition engine can infer two sets of web
services (indicated in Figure 6 as square textboxes in white):
(1) location service; (2) proximity calculation service (e.g.
buffer zones around the current location). In this episode,
geographical information services are used to support user’s
goal in determining a destination for the tour within the
given time limit.

The second episode describes the situation when the user
has examined the possible locations suggested by the system
and needs to decide on the destination to visit. The system
reasons on the state of the activity (based on the PlanGraph)
and determined that the most appropriate behavior is to
show a map of potential destinations by highlighting the places of interests according to the distance to current location. See the map in Figure 6. In order to generate this map, a request to ‘map presentation service’ is identified and executed.

While the user physically navigates in space, the system demonstrates helpful behavior by displaying a map of the route from current location to the destination. Again, a ‘map presentation service’ is to be requested.

The above scenario, although relatively simple, does demonstrate the evolution of the activity plan and its impact on the web service composition. The example suggests that composing web services requires a degree of adaptivity in order to be coupled dynamically with the situated activity.

V. IMPLEMENTATION

A prototype software agent, MyTour, is implemented in this study to demonstrate the feasibility of our approach in the motivating scenario. As outlined in Figure 8, the system is implemented with a modular architecture, including four major modules. The mobile client serves to monitor the ongoing interaction between the user and the system, capture all the meaningful inputs (e.g. user’s requests or location coordinates), and send them to the activity manager. The activity manager is the core component of the system, which maintains the PlanGraph structure, performs reasoning to keep the PlanGraph updated, and generates map-based outputs to the client. The knowledge base provides general knowledge that the activity manager needs in different phases of the reasoning process. The mapping component is in charge of interacting with underlying spatial information infrastructure.

A. Activity Manager

The activity manager is at the centre of the whole system design, which maintains the PlanGraph structure and performs associated reasoning algorithms.

PlanGraph is implemented as a dynamic data structure by several data objects (Figure 9). The overall PlanGraph object includes the root plan node and current attention focus. Each plan node includes attributes recording the parent node and the subsidiary plan nodes, which together allows the recursive traverse throughout the PlanGraph hierarchy. In addition, each plan node also includes a list of participating agents. We use two subclasses of the PlanNode class to represent two specialized types of plans: (1) ActionNode defines the properties of the action, such as the type of the action (e.g. single-agent/multi-agent, basic/complex). Each ActionNode also includes several slots to define the system’s beliefs about the corresponding action, as defined in Section III; (2) ParamNode models knowledge pre-condition of a plan as parameters to be identified and used.
1) **Reasoning Engine.** The reasoning engine in the activity manager organizes these reasoning processes in two levels. At a higher level, four module functions are implemented to control the overall workflow of the reasoning process: The Input Interpretation module; Plan Explanation module; Plan Elaboration module; and Response Control function, which correspond to the reasoning process discussed in section 3.2. At a lower level, each of these four modules is implemented as part of a logical programming engine to allow the dynamic inference behaviors. As discussed in Section 3.3, each step of the reasoning process is guided by different reasoning rules with dynamic sets of facts about the activity. To achieve this goal, we employ a knowledge engine Pyke [12] to enable both the forward-chaining and backward-chaining inference. In the Plan explanation module, forward chaining is used to update the system’s beliefs in the PlanGraph. The reasoning engine uses the PlanGraph to assert all the facts about the shared plan, and then activate plan justification rules to generate new facts about the shared plan. During the Plan Elaboration phase, the backward chaining process is used to derive further intentions and beliefs that the system may adopt from the initial commitment to the group activity.

2) **Knowledge Base** The Knowledge Base module stores different types of knowledge used in the system’s reasoning and behavior planning processes. We use a relational schema (stored in a relational database powered by PostgreSQL) to capture the knowledge necessary to (1) map user’s spoken language input to *semantic units* (actions, location, time, numbers, objects, etc), (2) choose *recipes* for actions, (3) discover appropriate geographical information services based on *metadata*, and (4) make behavioral decisions based on *adaptation rules*. Whenever semantic markup is needed, XML language is used.

**B Client-Server Interaction**

The system adopts a distributed architecture where the interaction between mobile clients and the activity manager is through the standard HTTP. Each time a mobile client registers a map service to the server, the server creates an instance of activity manager, which is in charge of modeling the activity with the mobile client and providing map displays. On the other hand, the assess to geospatial data services is also through HTTP, where the system can integrate map data from multiple data sources and provide on-the-fly vector data as well.

The client we currently use in the prototype system is built on the Android mobile platform. The system supports three forms of input from the client: (a) the selection of certain action from possible action lists, (b) free-text input, and (c) location information retrieved by the mobile device. The user’s input is encoded in a simple request XML document that is sent to the server. When the system receives a request from the client, it dispatches the request to the corresponding activity manager based on the unique client identifier. The activity manager follows the reasoning process to update the PlanGraph and return appropriate map responses to the client. The response message is encoded in an XML format describing the map content and styles (following the OpenGIS Web Map Context (WMC) specification), and is interpreted by the client to render the map.

**VI. DISCUSSION AND CONCLUSIONS**

In this paper, we have described a computational approach to modeling situated activities to contextualize mobile map service composition. Specifically, we adopt the SharedPlans theory to build the activity model, which has the capabilities to: (a) capture the internal mental attitudes of the users, (b) model the hierarchically structured actions in an activity that are intended and performed in some specific situational components (e.g. constrains, knowledge preconditions, intentional structure, and attentional state etc.); (c) and adapt to the ongoing development of activity as the shared plan evolves from partial to full. Implementation of the prototype system, *MyTour*, demonstrates the feasibility of the collaborative plan approach and the PlanGraph model to capture the dynamics of the usage context from the activity-centric perspective. Our work extends the AI planning approach of web service composition with the added flexibility and robustness of intention-based plans to achieve activity adaptive service composition.

This study can be further extended in several directions. First, our prototype system has not been deployed and tested with real user activities, and the validity of our method has not been tested yet. In order to know how well they work in real situations, a user-centered evaluation study is necessary. Second, the performance of the system depends on the quality of various types of knowledge built into the system, such as the domain knowledge about the actions and recipes, and cartographic knowledge to generate appropriate map displays. Therefore, a knowledge elicitation process that allows us to collect these types of knowledge from human experts/users in the application domain is planned for future study. In addition, the function of the input interpretation module of current system is very limited and only supports the lexical level interpretation.

In the future, we plan to explore the possibility of integrating multiple modals of user inputs (e.g. speech,