A PLAN EXECUTION MODEL FOR MOBILE PERSONAL ASSISTANTS

Incheol Kim and Huikyung Oh
Department of Computer Science, Kyonggi University, San 94-6, Yiui-Dong, Youngtong-Gu, Suwon, Korea

Keywords: Plan Execution Model, Smart Script System, Mobile Personal Assistant, Dynamic World, Events.

Abstract: This paper proposes a plan execution model suitable for dynamic mobile computing environments and presents a smart script system developed based on the model. The smart script system includes a script language which describes the task knowledge of mobile personal assistants and an execution engine which is designed to execute scripts dynamically according to task goals and environmental changes. In addition, this paper evaluates the usefulness and performance of the smart script system by implementing *Smart Reservation* as an application.

1 INTRODUCTION

Recently, with the rapid spread of smart mobile devices including smart phones and tablet PCs, there are many studies being conducted on mobile personal assistants that offer a wide range of convenient services for users (Gruber, 2009, Morley and Myers, 2004). Mobile personal assistants are assumed to perform the given tasks autonomously or semi-autonomously for their users, so they should be able to execute their own task plans properly responding to dynamics of the physical and information environments where they are working. To implement those intelligent mobile personal assistants, a high-efficiency reactive plan execution system should be developed (Ingrand and Py, 2009, Gregory et al, 2001).

This paper proposes a plan execution model suitable for dynamic mobile computing environments and presents a smart script system developed based on the model. The smart script system includes a script language which describes the task knowledge of mobile personal assistants and an execution engine which is intended to execute scripts dynamically according to task goals and environmental changes. Lastly, the usefulness and performance of the smart script system are evaluated by implementing an application called *Smart Reservation*.

2 PLAN EXECUTION MODEL

In this paper, the smart script system was developed based on the following plan model and plan execution model:

p = (ID, Name, Type, Body, Comment), where
ID = an identification number,
Name = a plan name,
Type = an event-driven plan, goal-driven plan, or command-driven plan,
Body = a set of actions such as $<a_1, \ldots, a_n>$,
Comment = a description about the plan.

The plan model **p** can represent three different plan types: Event-driven plans p_e are automatically executed when a particular event takes place; goal-driven plans p_g are intended to accomplish specific goals and command-driven plans p_c can be selectively and immediately executed anytime as needed.

The plan execution model (PEM) for the smart script system is defined as follows:

PEM = (P, S, W, U, I, N, T), where
P: A set of plans, $P = P_e \cup P_g \cup P_c$,
P_e = event-driven plans,
P_g = goal-driven plans, and
P_c = command-driven plans.
S: A sensor designed to sense the events that occur in an external environment,
$S : E \rightarrow 2F$, where E = a set of events and

Kim I. and Oh H..
A PLAN EXECUTION MODEL FOR MOBILE PERSONAL ASSISTANTS .
DOI: 10.5220/0003620602320235
In *Proceedings of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2011), pages 232-235
ISBN: 978-989-8425-78-2

F = a set of facts.

W: A set of facts representing the current status of an environment (i.e., a world model), $F \subseteq W$

U: Control commands associated with plan execution,
$u \in$ {execute_plan, stop_plan, activate_plan, deactivate_plan, ...}

I: An interpreter designed to decide which plan to execute,
$I(W, u, P) = p$, when the plan $p \in P$ has its preconditions satisfied with the current world model "$W \models PRECOND(p)$" or is associated with a user command "$u \in U$."

N: An intention structure designed to manage the process in which a decided plan is actually executed,
$N(p)=SUCCESS(DO(a_1))?\ N(p')\ :\ STOP(p)$,
when $BODY\_SEQ(p)=<a_1, a_2, ..., a_n>$,
$BODY\_SEQ(p')=<a_2, a_3, ..., a_n>$,
Action $a_k \in A$

T: An effector designed to change an environment by completing each of actions which constitute a plan, $T : A \rightarrow E$

PEM provides rich, expressive plan representations, a wide range of useful plan execution semantics, reactivity to environmental changes, support for user interaction in runtime.

3 SMART SCRIPT LANGUAGE

A smart script language was designed based on the plan and execution models mentioned above. The smart script language allows users to use a wide range of built-in variables and other user-defined variables. In this language, the status of an environment is expressed as a set of facts.

Fact ::=" (" Variable Value ")"

In other words, a single fact is expressed as a variable and its current value, like (%SMSRA "02-123-3456"). The set of actions available in the scripting language consists of control actions and effective actions. The control actions which determine the order of execution include goal actions, script actions, IF actions, ENDIF actions, WHILE actions, ENDWHILE actions, WAITUNTIL actions, WAIT actions, SUCCESS actions, FAIL actions and SET actions. The goal action is to add a new subgoal; the script action is to request the execution of a new script; and the SET action is to assign a value to a user-defined variable.

The smart script language provides a variety of effective actions including speaking (SPEAK), making a phone call (CALL), sending a text message (SENDSMS), and playing a music file (PLAYMUSIC).

Three different types of script (command-driven, event-driven and goal-driven) can be defined in the smart script language. The command-driven scripts mean those scripts that users are able to execute themselves.

```
Command-driven Script
  ::= "(" "script"  Script-Name
         [":context"  Condition-List]
          ":body" Action-List
         [":comment" Description] ")"
```

The event-driven scripts are automatically executed in response to a specific event taking place in an external environment.

```
Event-driven Script
  ::= "(" "script" Script-Name
          ":precondition" Condition-List
         [":context" Condition-List]
          ":body" Action-List
         [":comment" Description] ")"
```

The goal-driven scripts are defined as those scripts that can be automatically selected and executed to accomplish a goal, once the goal is set out by a user or a higher-level script.

```
Goal-driven Script
  ::= "(" "script" Script-Name
         ":goal Goal
        [":precondition" Condition-List]
        [":context" Condition-List]
         ":body" Action-List
        [":comment" Description] ")"
```

The smart script program is composed of three different components: goal declaration, fact declaration, and script declaration.

```
Script Program ::=
         [ Goal-Declaration ]
         [ Fact-Declaration ]
           Script-Definition

Goal-Declaration ::=
      "(" "goals" Goal-Action-List ")"
Fact-Declaration ::=
      "(" "facts" Fact-List ")"
Script-Definition ::=
      "(" "scripts" Script-List ")"
```

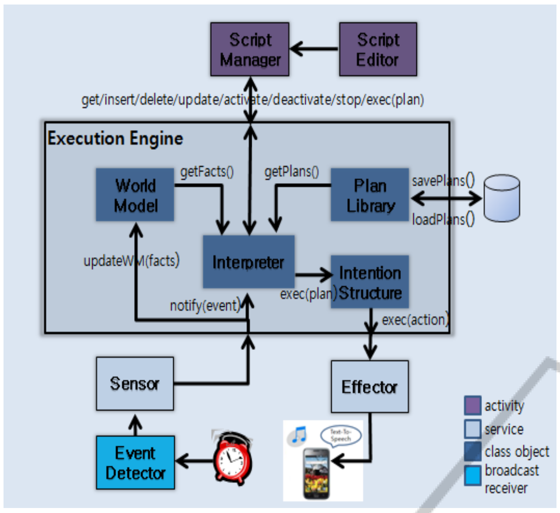
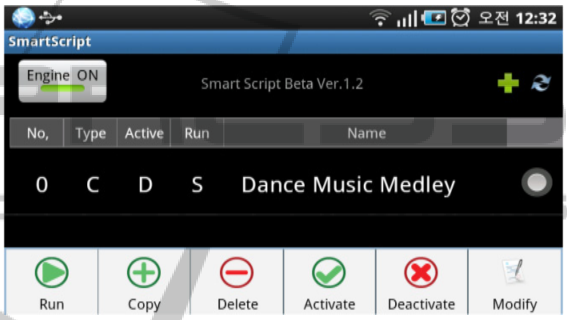


Figure 1: Architecture of the smart script system.

4 SMART SCRIPT SYSTEM

As shown in Figure 1, the smart script system consists of the *Execution Engine* which controls the automatic execution of scripts, the *Sensor* which receives diverse event and status information from environments, the *Effector* which is responsible for the execution of individual basic tasks or actions, the *Script Editor* which allows users to edit or modify scripts, and the *Script Manager* which allows users to monitor and control the execution of scripts in real time. The *Sensor* detects a variety of events that may occur in smartphone environments, including text-message arrival, phone ringing, WiFi detection and schedule notification, and works to update an internal *World Model* of the *Execution Engine* based on new facts associated with those events.

The *Execution Engine* responsible for the execution of scripts has internal components with different roles, such as *World Model*, *Plan Library*, *Interpreter*, and *Intention Structure*. The *World Model* manifests the current status of the system in which scripts are being executed, by saving and updating the environmental status or user-entered data detected by the sensor or data on the results of each basic task/action executed. In the *Plan Library*, user-defined scripts are stored in an execution waiting status, and whenever necessary, it stores scripts in a database or brings up the stored scripts from the database. The *Interpreter*, based on the facts of the *World Model*, checks whether the preconditions of individual event-driven scripts in the *Plan Library* are satisfied or whether each goal-driven script is matched with a user-supplied task goal and plays the role of choosing a script to be executed next. The *Intention Structure*, with the script chosen by the *Interpreter*, performs script-intended tasks by executing in sequence the basic tasks/actions described in the body of the script. The *Execution Engine* has been designed as an Android service so that it can start the quick execution of scripts whenever the script-associated events occur. While the *Execution Engine* executes the logical decision-making process of the smart script system, the *Effector* executes each of basic tasks/actions actually as requested by the *Execution Engine* with existing s/w tools such as web browsers, calendars and email programs or its own programs implemented with Android API



No,	Type	Active	Run	Name
0	C	D	S	Dance Music Medley

Figure 2: Screenshot of the script manager.

The *Script Manager* provides the main user interface of the smart script system. Users can execute one of the command-driven scripts through the script manager, suspend running scripts, or activate/inactivate event-driven scripts. In this way they can control the scripts in real time. In addition, with the help of the *Script Editor*, they can create copy, modify or delete scripts. Taking into account the limited size of smart phones' screen and keyboard, the *Script Editor* has been programmed to minimize users' keyboard entries by applying graphic icons and menu selections to most editing procedures. Figure 2 shows the screenshot of the *Script Manager*.

5 APPLICATION

In order to examine the performance and usefulness of the smart script system developed in this work, *Smart Reservation* has been created based on this system. *Smart Reservation* is an application which finds a nearby restaurant and makes reservations for smart phone users in semi-autonomous way. *Smart Reservation* can get the user's location information from GPS to search for nearby restaurants and have

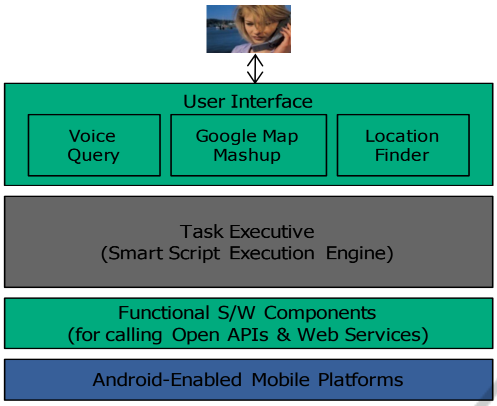

User Interface
Voice Query
Google Map Mashup
Location Finder
Task Executive (Smart Script Execution Engine)
Functional S/W Components (for calling Open APIs & Web Services)
Android-Enabled Mobile Platforms

Figure 3: S/W organization of *Smart Reservation.*

the user's voice queries entered. This application translates the user's voice query into a SPARQL form of semantic query and then sends the query to a remote restaurant information server via web service. Based on the search results from the information server, it marks restaurant locations on the map using Google Maps mashups and even helps make online reservations at the user's desired restaurant.

Smart Reservation, as illustrated in Figure 3, has been developed based on the smart script system, and accordingly the key task knowledge of *Smart Reservation* is expressed as scripts which are executed by the smart script execution engine. We implemented unique some state variables and actions for *Smart Reservation* using an open API and web service calls. Additionally, we created a new user interface which supports voice-based interactions in view of user convenience. Figure 4 shows a screenshot of *Smart Reservation*. Even though the implementation of *Smart Reservation* is currently limited to the use of information on restaurants in Insa-Dong, Seoul, it was enough to ensure the usefulness and performance of the smart script system as a framework which enables a variety of mobile personal assistants to be very effectively developed.

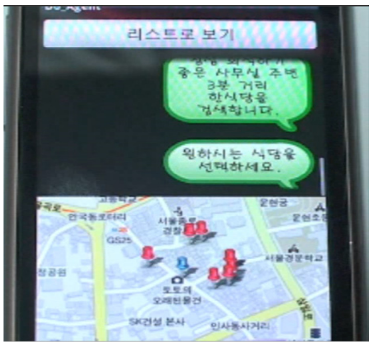


Figure 4: Screenshot of *Smart Reservation.*

6 CONCLUSIONS

This paper proposed the plan execution model and presented the design and implementation of a smart script system based on this model. Due to reactivity and goal-orientation of the underlying model, the smart script system can respond very quickly to changes in mobile computing environments and provides the robust framework for implementing various mobile personal assistants.

ACKNOWLEDGEMENTS

This work was supported by the GRRC program of Gyeonggi province.

REFERENCES

Gruber, T., 2009. Siri: A Virtual Personal Assistant, *Keynote Presentation at the Semantic Technology Conference(SemTech-09)*.

Morley, D., Myers, K., 2004. The SPARK Agent Framework, *Proc. of the AAMAS-04*, pp. 712-719.

Ingrand, F., Py, F., 2009. *Proc. of the 4th Workshop on Planning and Plan Execution for Real-World Systems, ICAPS-09*.

Gregory, N. M., et al, 2001. IDEA: Planning at the Core of Autonomous Agents, *Proc. of AAAI-01*.