Authentication and location control via RFID analysis

Khalid Agourare
LACL, Paris 12th university
61 avenue du General de Gaulle
94000 Creteil, France
khalid.agourare@univ-paris12.fr

Fabrice Mourlin
LACL, Paris 12th university
61 avenue du General de Gaulle
94000 Creteil, France
fabrice.mourlin@wanadoo.fr

Abstract

To develop and demonstrate accurate allocation and placement on network, we implemented a flexible location framework which is able to use RFID sensors for authentication and positioning adequate software environment. This application is particularly useful in an evolving network where a workstation is not dedicated to a specific use or user. Each user has own RFID badge and each workstation also. When a new user wants to access to network, first, he has to be recognized by the front server. The server loads information about user work context and books a free workstation for this user. This work context is export to the workstation by the use of mobile agent. The new user will have to connect on the particular workstation and will have access to all his data and application. Mobile agents are also controllers and observe and check what the user does.

1. Introduction

RFID systems have been used quite extensively for various types of applications which involve tracking [1], identification [2], access management [3], etc. These applications have to exchange with different RFID hardware devices such as readers and tags, to get information.

RFID provides an efficient way of automatically identify various objects. This property of the RFID devices has enabled it to be used in many applications concerning identification and tracking. One of the most widely used applications is the access control systems, where RFID based plastic cards are used to identify and authenticate the card-holder's entry to the facility. The RFID systems are extensively used in the warehouses and shops for the supply chain management [4, 5], inventory management [6] and movement management [1, 7]. This has led to huge increase in the efficiency of the warehouse operations and keeping the optimum inventory in the stores. Here we discuss some RFID based applications.

The context of our work is the management of access of computing network. This resource is precisely controlled because it is limited, costly and its planning is supervised by administrator. The basic concept is numerical analysis project which uses resources (processor, memory, files, etc.). Input data files are huge and are shared between projects. So, a numeric project corresponds to time periods and a user has right to work on one or more projects. However a user can have access only to several source files, input data and his actions are to be monitored.

These main constraints lead to create an identification process for user and material resource. This is why; we decided to assign RFID car for each user and a RFID tag for each workstation. But every resource cannot receive a transponder and an antenna. Also, our control approach has to be completed with a software controller. In this paper, we present our management strategy of computing resources through the use of RFID tags. We use two kinds of RFID tags, passive for the user and active for the material. We also present features of RFID tags that are useful for the development of our management application. Our experience about mobile agent technology [8, 9] allows us to couple RFID identification with mobile agent exportation to a workstation, where the user is assigned.

The rest of this paper is structured as follows. In Section II, we detail the requirements for an RFID infrastructure and our programming choices. Section III provides an overview of mobile agent technology and outlines the constraints imposed by the RFID features. In Section IV, we describe our RFID platform called MARFIDE. We continue by presenting some sample applications that were developed with MARFIDE in Section V. In Section VI, we show how our implementation addresses the application needs and technology constraints, and we review limitations of our implementation and present future work. Then, we conclude the document.

2. RFID programming approach

RFID transmits the identity of an object as a unique serial number. This identity is stored in the tag chips and can be retrieved by the Readers. The components of RFID are: tag, reader. Tag, also called transponder is a small device which contains a microchip. The chip is used to store the data. The tag can be programmed with specific items of information, such as an ID or serial number or a user data. The tag identifies itself by transmitting signals to the reader. There are two types of Tags namely: Active Tag, Passive Tag.

A reader consists of one or more antennas that emit and receive radio waves. The micro chip circuit present in the tags is powered by this signal. When the tags enters into the radio wave field, it transmits its unique information (unique serial number or user data) to the reader by modulating the signal. The reader converts the signal obtained from the tag as digital information and passes to the applications.

The first role of RFID card is to identify user. The success of this step leads to look up information into database and then to export agent through the network. We have already worked about mobile agent and we used Java language to realize mobile code. Our previous experience involves the selection of Java language for this new development. Java System RFID Software is an RFID infrastructure framework that integrates data and RFID-enabled devices into application systems.

2.1. Java and RFID: an open approach

This framework facilitated asset tracking, enforce policy compliance, detect tampering, and prevent anomaly It allows developer to gathers and filters information from RFID readers; exchanges this information as XML messages; stores the information to relational databases; and manages RFID devices, events, and components.

Sun Java System RFID framework consists of four modules. First, a RFID event manager is an event processing framework that manages data coming from all readers in the network. Secondly, a RFID configuration manager is a GUI application which allows developers to specify the set of devices connected to the RFID Event Manager. Then a RFID management console is a Web-based application used to modify various read and write attributes of the RFID Event Manager. Finally, a RFID information server is a J2EE application that facilitates capturing and querying EPC-related data and provides interfaces for back-end systems.

The RFID Event Manager framework is a Jini-based event processing framework that facilitates the capture, filtering, and storage of EPC events generated by RFID devices. The Jini services of the RFID Event Manager framework are managed through Rio server, an open source container of a component model called Jini Service Beans (J.S.Bs).

The RFID Event Manager consists of a management application called "Control Station" and one or more workload handlers called "Execution Agents." Execution Agents register with the Control Station application, which uses them to gather and filter RFID information. Each Execution Agent provides one or more specialized Web services called "reader services." A reader service communicates with the RFID device, processes the information according to a configuration object, and communicates an event client.

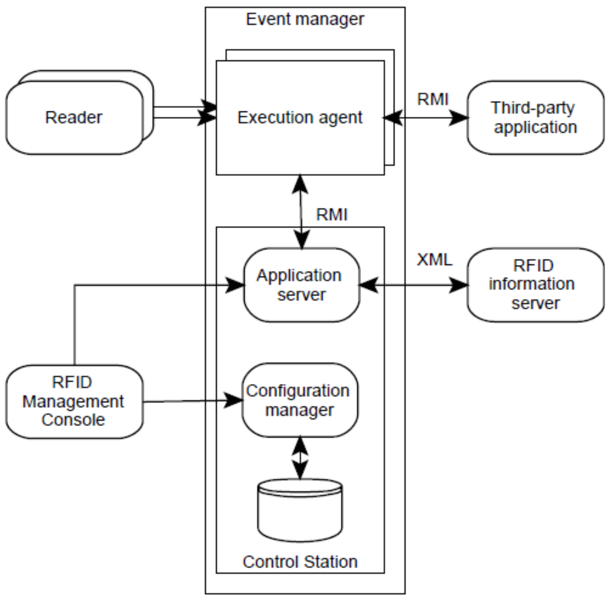

Figure 1. Event manager relationships

2.2. Configuration

Configuration Object components generate events to communicate with each other. Every component is an event producer and every component can register itself with other components to consume their events.

The RFID Information Server is a J2EE application that captures and queries EPC-related, low-level data and translates it into higher-level business data. RFID Information Server is typically deployed on a different machine than the RFID Event Manager. The RFID Information Server is built on Jini 2.1 (Java intelligent network Interface), Project Rio 3.1, and the Sun Java System Application Server 8.1. The impact of Jini framework is essential because it is the basis of our approach of mobile agent.

3. Mobile agent technology

Mobile agents are defined as active objects (or clusters of objects) that have behavior, state and location. Mobile agents are autonomous because once they are invoked they will autonomously decide which locations they will visit (their roadmap) and what instructions they will perform (their task). This behavior is either defined implicitly through the agent code (see e.g. [8]) or alternatively specified by an itinerary at runtime modifiable (see e.g. [9]). Mobile agents are mobile since they are able to migrate between locations that basically provide the environment for the agents' execution and represent an abstraction from the underlying network and operating system.

With the properties printed out above it has been often argued that mobile agents provide certain advantages compared to traditional approaches as the reduction of communication costs, better support of asynchronous interactions, or enhanced flexibility in the process of software distribution.

The employment of mobile agents has been particularly promising in application domains like information retrieval in widely distributed heterogeneous open environments (e.g. the World Wide Web), network management, electronic commerce, or mobile computing.

Several implementations already exist but few of them hide technical features from their environment. Data exchange is also a strong constraint in agent community; data type has to be preserved from the sender to the receivers. These main constraints helped us to select a mobile agent framework. Because there is no framework which respects these features, we developed our own framework which was presented at the ESM 2005 conference [10].

3.1. Mobile agent framework with Jini

Our mobile agent framework consists of two main components. The first component is mobile agent (Figure 2 (A)); that is, entities with some task to do. A real application contains several kinds of mobile agent. Each kind has its own task and each agent has its own road map. The task can be depending on the location where the agent is.

The second component is the mobile agent host(s), the service that provides the mobile agents' execution platform. In a distributed environment, we can have one-to-many agent hosts as well as one-to-many agents. To be an active agent platform, a given node in the system must have at least one active agent host. Figure 1 describes the framework components.

Both agent and agent host have to publish their availability in public registry (Figure 2). This publication contains a record of data, each field of this record qualify a facet of the service: signature, scope, exception case, permission control, etc.

To sum up, each node of the network of workstations possesses at least one agent host and a public registry at the beginning of an execution. The agent host has to publish into the local public registry its ability to receive agents. This service filters specific kinds of mobile agent. It is also possible to overload the reception service. Each of the services describes a precise process of reception (control, propagation, or eventually measure).

Our framework is based on Java language and Jini technology [10]. The Jini networking system is a distributed infrastructure built around the Java programming language and environment. Jini is the name for a distributed computing environment, that can offer ``network plug and play''. A device or software service like an agent can be connected to a network and announce its presence, and clients that wish to use such a service can then locate it and call it to perform tasks [11]. Jini can be used for mobile computing tasks where a service may only be connected to a network for a short time, but it can more generally be used in any network where there is some degree of change.

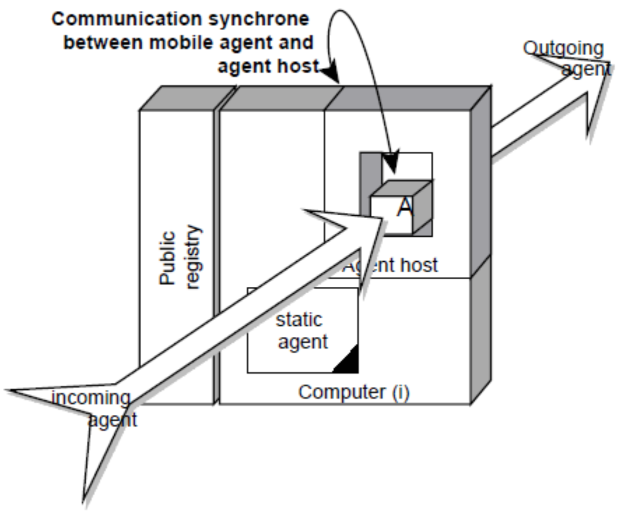

Figure 2. agent platform architecture

The basic communication model is based on the semantic model of the Java RMI (Remote Method Invocation) system, in which objects in one Java virtual machine communicate with objects in another by receiving a proxy object that implements the same interface as the remote object. This communication model is the core feature for moving agents. The proxy object deals with all communication details between the two processes. The proxy object can introduce new code into the process to which it is moved. This is possible because Java byte codes are portable, and it is safe because of the Java environment's built-in verification and security.

To this underlying communication model the Jini system adds some basic infrastructure and parts of a programming model. The infrastructure provides a mechanism by which clients and services can join into the Jini network while the programming constructs encapsulate mechanisms that allow reliable distributed systems to be built. Java provides the Jini system with a mechanism to move mobile objects, including their code, safely and efficiently from a service to a client of that service. The Java type system forms the basis for identifying services, and its polymorphic nature lets it treat requests for service as requests for something that implements at least a certain type, although the service might offer more. However, the requirement for the Java language and platform is only at the network level; mobile agent programmers can use our Mobile Agent

Framework to implement a mobile agent system that can live in the Jini environment.

3.2. RFID impact

The first role of mobile agent happens when a user wants to have a connection to the network. When the user RFID tag is read by the server, his identifier is checked regarding a database. Next, his working features are extracted for the configuration of a mobile agent. Then a workstation is assigned for the user connection and because there is an agent host on that workstation, the mobile agent is received. On its arrival, it configures the workstation for future user activities and then is waiting for the connection of the user.

This user has to type his information connection (login, password, key code). These data are checked by the mobile agent and data are exchanged with the RFID tag of the workstation. This marking corresponds to the start point of a user working period. When the user will stop his connection, the spent time for the user activities will be computed with reference to the start point [12].

We use a mobile agent community for the control of user activities. Each class of agent has its own task, for instance, the initial exportation of working environment, the control of connection data, the cleaning of the workstation at the end of connection, etc. Our framework is open and new mobile agent can be developed and used by any agent host. For the communication with RFID devices, we developed new classes of mobile agent depending on the communication protocol of the devices. This work is a part of the framework MARFIDE.

4. MARFIDE Platform

MARFIDE stands for Mobile Agent for RFID Exchange. This work is an extension of a project called JIMA (JIni mobile Agent) which we developed for the effective software management on a network. A mobile agent community realizes the observation of all the activities of used software. A new set of agents is deployed when new software installed over the network. These agents give us information about the load, the use, as wall about what it is not used [11]. MARFIDE completes this approach with the tracking of users. Now, we cover not only the software use but also the user activities and their dependencies. First we defined deployment architecture and the key constraints led the software architecture of our control software.

4.1. Deployment architecture

We consider (figure 3) an example with three workstations and a server which is able to scan a user card. The server contains mobile agent server and a public registry where all the available agents are declared. Each user card has a passive tag which uses the radio frequency from the reader of the reader to transmit its signal. The passive tag has data permanently burned into the tag when it is made, and each user has own card. It takes part into the first authentication step [13].

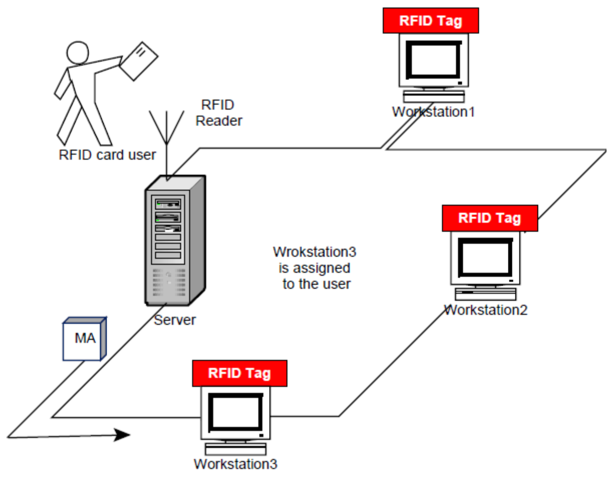

Figure 3. material architecture.

On a workstation, an agent host is running with a local public registry. The remote features of the agent host are declared into this registry (like figure 2). It has also static agent to exchange data with the local RFID tag of the workstation. This is an active tag which has on-board battery for power to transmit their data signal over a greater distance and power random access memory (RAM) giving them the ability to store up to 32,000 bytes of data. Two kinds of data exchange are useful: first between the passive tag of the user card and the reader of the server, secondly between the active tag of the workstation and a local static agent.

Now we explain how RFID gives workstations the capability to track not only what but also where an item or person is.

4.2. Handling Application-level RFID events

Sun Java System RFID offers an abstraction of Application-level Events (called ALE) defined by the EPCglobal specification in the specific interface, called com.sun.autoid.ale.AleAPI. Classes implementing this interface embody support for handling Application-level Events. Sun provides a default implementation of the AleAPI interface in the package com.sun.autoid.ale.client called, ALEClient class. Our application needs to be programmed to handle the XML messages that are received containing the requested information. Also, this client class is used to receive and transform an XML stream.

The ALEClient class constantly searches for all readers on the system and maintains an up-to-date list of physical and logical names. The code just below, illustrates how to use the ALEClient class to support the ALE specification. This code connects to the ALE service specified by its url parameter: This class is the core use of the reader of the server (figure 3).

```
ALEClient aleClient = new
ALEClient("http://server:80/rfidWS");
```

Another important Java class is the Java ReaderClient APIs to control devices, program RFID tags, read and write user memory and tag identification on RFID transponders (tags) using the Java library bundled with the RFID Software 3.0. Our application communicates directly with the RFID Event Manager by using Java RMI (Java Remote Method Invocation) without the need to convert between protocols and data representation.

We chose to use ALE to obtain tag information, and then use the device access web services to get user data and program tags. We wanted to mix and match web service and Java APIs. For example, we use the Java ReaderClient API to communicate with RFID devices in our local network and use the ALE Java library or the device access web services to communicate with an RFID Event Manager in a remote network across the world.

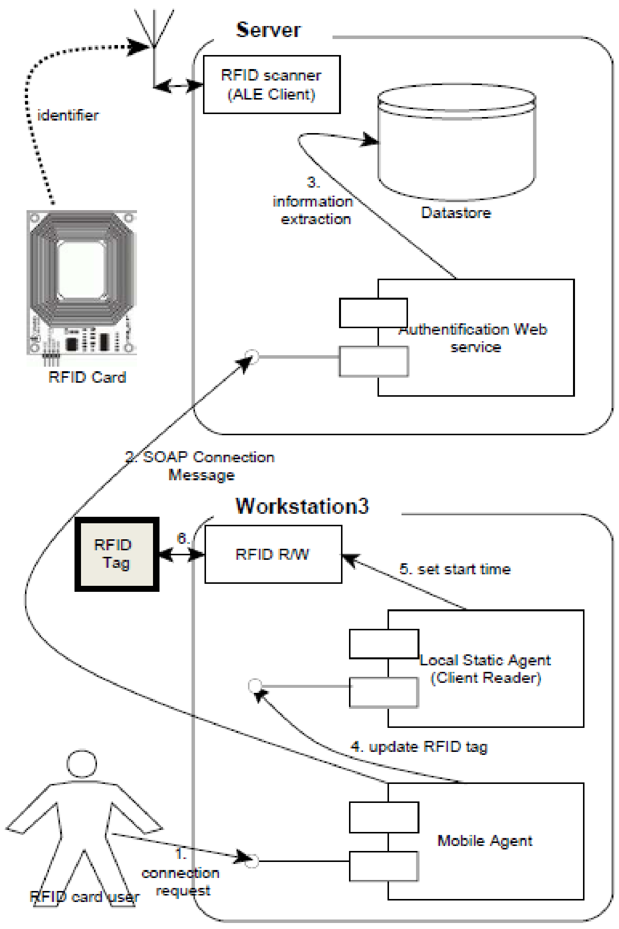

Figure 4. Configuration during connection

Once, the user is in front of the chosen workstation, he can enter his information and work about his project. User actions are now under control of the mobile agent. This one comes from the server with the features of user context. A set of permissions is built per project and a member of the team has permissions which give rights to do activities on that project.

Because permissions can be added during users' activities, the set of user's permissions is checked constantly. This is realized by the use of web services. So, on the server, there are a web server and a set of web services. All of them are developed under JAX-RPC (Java API for XML-based RPC).

The Sun Java RFID Software implements an ALE Web service using JAX-RPC. This service acts as an intermediary to a Jini RMI service contained within the RFID Event Manager. Requests and messages sent to this Web service are referred to as "report requests" and "report messages" respectively, and are implemented as Plain old Java objects (POJOs). The ALE Web service supports three basic modes of interaction with clients: subscription, polling predefined ECSpecs, and polling a new ECSpec. This allows asynchronous data exchange. We use JMS (Java Message Service) API for the definition of each mode. The subscription mode is used for the administration of our application. This means to build a map of all activities on the network.

4.3. Administration RFID system

Monitoring tasks are essential for a precise management. The following monitoring tasks on a regularly scheduled basis will assist administrators in keeping MARFIDE server applications and infrastructure operationally ready

The daily monitoring tasks are: the review of all open alerts, the control that the server computer is communicating with all the workstation, the use of the administration console to investigate orchestration, port, and message failures. It provides access to the current real-time state of the system, accessing data in the message box database. It is possible to observe all service instances such as orchestrations, ports, and messaging, along with their associated messages. A look into the message box database allows viewing the current data and the real-time state of the system.

Weekly Monitoring Tasks are the review of the event logs. The reason for this task is to prevent issues, such as service interruption. However, some of these errors can indicate a bigger issue (for example, too many workstations or an excessive load server, performance issues on the SQL database, etc).

Others tasks can be done as-needed such that the modification of the permission rules to customize a project by replacing a member or the declaration of a new secure policy. Other modifications are about the material architecture (new workstation, new service, new project, etc.). Each task is defined by a set of classes and MARFIDE is open and a new developer can define new tasks through a new development. This is a way to interface an external tool such that performance tool or simulation tool for a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types

5. MARFIDE applications

Our first application with MARFIDE framework is the tracking of user activities. Other domain can be concerned by this kind of control such that document access. While the document is moving around there was often no way the requester, could easily know where it was, who was (or was supposed to be) working on it at any given time, or when it might be completed.

5.1. User activity control

The basis for wearable computing is the detection of the user's context. With the appropriate RFID sensors and algorithms, context-based information delivery becomes feasible: the user receives adequate information automatically when he needs it, without having to explicitly ask for it. Context-based information delivery can improve productivity in some work environments, for instance for test sequences by loading particular input data. Next figure highlights the states of the working user context. A local access controller checks user permissions for all user commands.

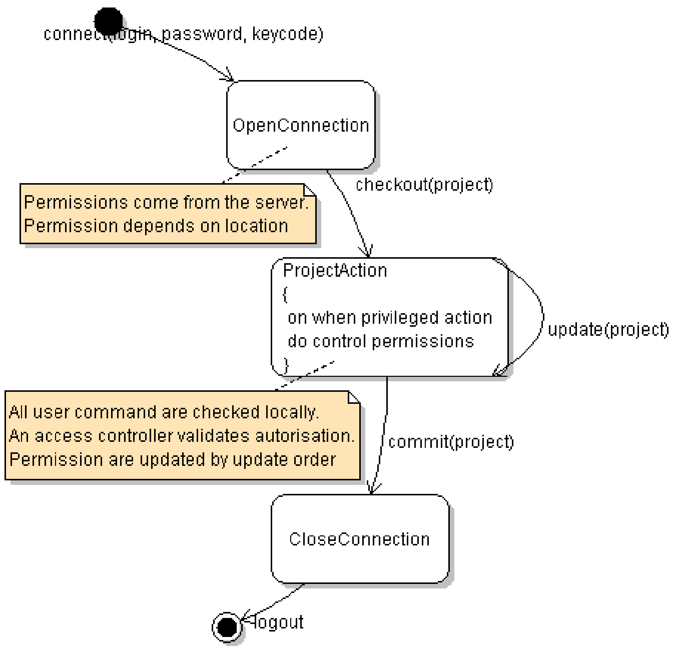
connect(login, password, keycode)
OpenConnection
Permissions come from the server.
Permission depends on location
checkout(project)
ProjectAction
{
on when privileged action
do control permissions
}
update(project)
All user command are checked locally.
An access controller validates autorisation.
Permission are updated by update order
commit(project)
CloseConnection
logout

Figure 5. user workspace statechart

Permission refers to an action that the application is allowed to perform. Each permission consists of a type, that indicates the resource type this permission refers to, a name that specify the resource this permission refers to, and the action that can be performed on this resource. As an example, permission could be: (marfide.security.ProjectPermission, 129.175.*.*, project1). This permission allows the Java application to set up a project connection with the remote host 129.175.*.*.

Permission depends on the project, the user and also on a precise location. When permission is missing, an exception is raised and anomaly is traced into a local log file. Next a mobile agent creates an alert and propagates it to the server. This messenger is useful if the user is connected on other workstations, this can involve the closing of every connection and others alerts.

Our access controller is history based because it does not simply allow the execution of an action by exploiting authorization rules that take into account only static factors, but it monitors the behavior of the project user, i.e. the sequence of actions performed by the project. Hence, to decide whether an action can be executed, the whole trace of execution of the application is evaluated. In this way, some dependencies among the actions performed by the user can be imposed by the policy.

Team members can understand through local log files, the work they are to perform and how that work fits into the road map of the project. This information helps them make good decisions in their day-to-day work. Local log files are collected at the end of the work session. The whole content is moved from the workstation to the server where it is parsed and saved. Its format is checked by the use of an XML schema which allows us to transform easily the structure to another one. This is essential because a same project can be used by another user on another workstation and a first activity can have consequences onto another one. The first repository is placed on the server. This means, project lifecycle is managed only on a specific computer.

5.2. Project activity control

Team members get more done if they can easily find the information that they need because it was clearly outlined from the start. Good communication also prevents duplication of effort, which hurts project schedules. So, each new member is declared on the server and user role in the project corresponds to permissions. So resources, source files are available under control of this user role.

When team members have the information they need, they make fewer mistakes. When problems do arise, informed team members can quickly identify them and collaborate to fix them. This kind of alert is created on the workstation where the user works. A mobile agent moves with this alert to the server where the alert is treated. This means change of a project configuration (state MemberManagement on figure 6), change of member configuration such that suppression of permissions, etc.

By studying information about every project that a company executes, project teams can copy the best practices while avoiding the mistakes of previous projects. This activity is done by analysis of log files which belong to a project. A rule engine applies practice rule to detect global anomalies. Then, project managers can limit the impact of complications by identifying both potential risks and what can be done to manage them.

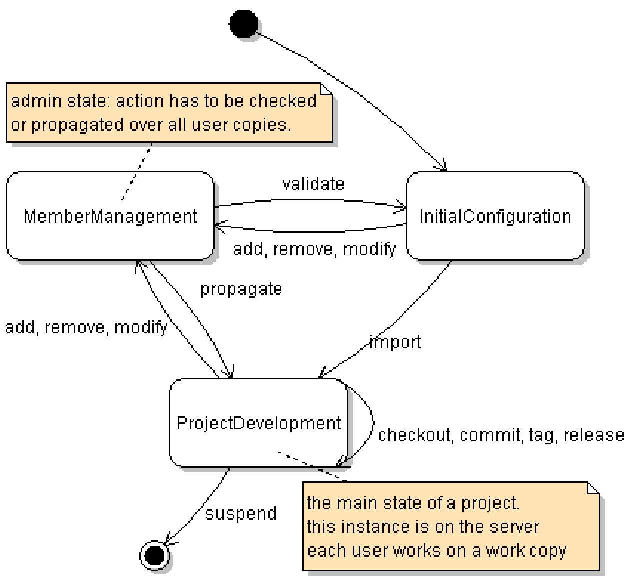
admin state: action has to be checked or propagated over all user copies.
validate
MemberManagement
InitialConfiguration
add, remove, modify
propagate
add, remove, modify
import
ProjectDevelopment
checkout, commit, tag, release
suspend
the main state of a project.
this instance is on the server
each user works on a work copy

Figure 6. user workspace statechart

A focus on collaboration and communication make user a more effective project member and help all team members and stay focused on results. We work on

5.3. Workflow approach based on mobile service

The operations on software project, we described in previous sections can be compared to document workflow except that these operations have impact on to the resources of the project. For instance, the input data are key resources for numerical application. Because these data are only available during a short period, we developed TimerPermission class to control the accesses.

Mobile agent is a key feature in our framework. A mobile agent is a composition of computer software and data which is able to migrate from one workstation to another autonomously and continue its execution on the destination computer. More specifically, our mobile agents are processes that can transport their state from one environment to another, with its data intact (for instance an alert or a project command), and be capable of performing appropriately in the new environment. We can compare mobile agent concern with web service. In the first case, the instructions or code move over the network. In web service case, these are data which are encoded into XML format and then move over the network.

The underlying mechanisms are quite similar. Movement is evolved from RPC methods. Just as a user directs an Internet browser to "visit" a website (the browser merely downloads a copy of the site or one version of it in the case of dynamic web sites). Mobile agents react autonomously to changes. Multiple mobile agents have the unique ability of distributing themselves among the hosts in the network to maintain the optimal configuration for solving a particular problem such that a new workstation, a change in the interconnection. If a workstation is being shut down, all agents executing on that machine are warned and given time to dispatch and continue their operation on another workstation in the network

6. Main results

The project monitoring means to keep a careful check on project activities over a period of time. All collected data are used by a project manager to provide boards. To work to its full potential, any kind of project needs to set out rules and constraints. Then our monitoring system should be worked out to keep a check on all the various activities, including project management, resource availability. This helps project staff to know how things are going, as well as giving early warning of possible problems and difficulties. We build representation from XML data and convert them to SVG format (Scalable Vector Graphic).

6.1. User activities

Our project manager client allows building graphical views about each project. These data comes directly from concrete activities of team member. Several kinds of views can be built for user per project during a time period (figure 7). More details can be represented if project operations are filtered; it means update, checkout, etc. Then, complex graphics is obtained and the analysis can be possible by a project manager.

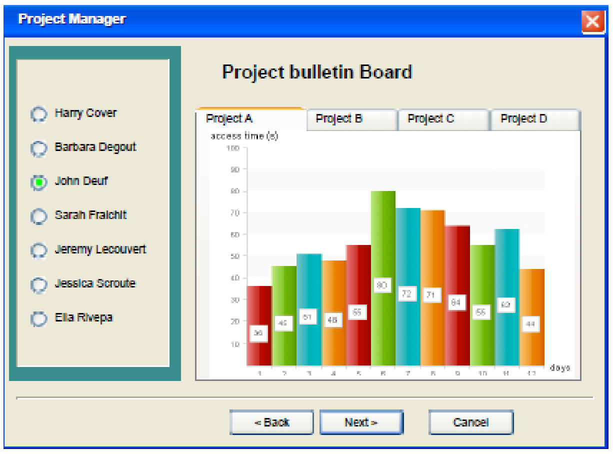

Figure 7. project board per user

User activity metric can be computed depending of operation sequences length or operation frequency, its goal is to improve the team activity and to create new management rules.

6.2. Project metric

Measurement is an essential element of project management. Project activity metric can be computed, its goal is to improve the team activity. They should be visible at least to the entire team, and perhaps beyond [15].

A team is more than a collection of individuals. Efficient teams create synergy, which enables them to accomplish much more than the sum of individual efforts. But for inefficient teams, even the smallest problem can have dramatic negative effects. Therefore, it is important to continually monitor the team

Primary measurements are about time, and size, for instance how much time is spent on different project operations or on specific commitment. A more complex question could be: are our process and application of software development techniques effective? To build an answer to these questions, data transformation has to be built and graphical observer also.

All the views are created in a SVG format because we want to have available observation from the server on client workstation. This data format allows team member to download view easily at any time during the lifecycle of the project.

7. Conclusion

In novel RFID application domains, such as project management, there are many RFID readers distributed across the network. The data the readers capture need to be disseminated to a variety of workstation. This introduces the need for an RFID infrastructure that hides proprietary reader device interfaces, provides configuration and system management of the reader devices, and filters and aggregates the captured RFID data.

In this paper, we discuss real application requirements in detail. We also contend that the characteristics of passive RFID technology are useful for the user but active RFID technology is better for the material. The paper shows that the current MARFIDE implementation, which is based on a set of specifications developed by the EPCglobal community, addresses the majority of the application requirements. Our work also discusses limitations of the existing implementation. We highlight the role of mobile agent for the data propagation and then mobility is a key feature to adapt a task execution in an environment which is not stable. Finally the data collection allows building enough data on the server which is the source of a set of transformations. Each transformation provides a graphical view or bulletin board for a project manager.

References

[1] A. Brewer, N. Sloan, and T.L. Landers. Intelligent Tracking in Manufacturing. Journal of Intelligent Manufacturing, 10:245□250, March 1999.

[2] H. Vogt. Multiple object identi□cation with passive RFID tags. IEEE International Conference on Systems, Man and Cybernetics, October 2002.

[3] M.R. Rieback, G.N. Gaydadjiev, B. Crispo, R.F.H. Hofman, and A.S. Tanenbaum. A Platform for RFID Security and Privacy Administration. Proceedings of the 20th conference on Large Installation System Administration Conference-Volume 20, pages 89-102, December 2006.

[4] Z. Asif and M. Mandviwalla. Integrating the Supply Chain with RFID: A Technical and Business Analysis. Communications of the Association for Information Systems, Volume 15:393-427, 2005.

[5] M. Karkkainen. Increasing Effciency in the Supply Chain for Short Shelf Life Goods using RFID Tagging. International Journal of Retail & Distribution Management, pages 529-536, October 2003.

[6] J.H. Bowers and T.J. Clare. Inventory system using articles with RFID tags, 1999.

[7] Federal Trade Commision. Radio Frequency Identification: Applications and Implications for Consumers. Technical report, Federal Trade Commision, March 2005.

[8] R. S. Gray. "AgentTcl: A Transportable Agent System", Proc. CIKM'95 Workshop on Intelligent Information Agents, 1995.

[9] Fowler, Martin, Refactoring: Improving the Design of Existing Code. Addison Wesley, 2000.

[10] Maamoun Bernichi and Fabrice Mourlin, "A New Behavioural Pattern for Mobile Code ",In ESM 2005, University of Porto, Porto, Portugal, 24-26 October 2005

[11] Maamoun Bernichi et Fabrice Mourlin, "Java mobile agents for monitoring mobile activities ", In Eurocon'05 conference, Serbia & Montenegro, Belgrade, November 22-24, 2005

[12] B. Fabian, O. G¨unther, S. Spiekermann, Security analysis of the object name service for RFID, in: Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2005.

[13] Y Hu, S Sundara, T Chorma, and J Srinivasan. Supporting RFID-based item tracking applications in oracle DBMS using a bitmap datatype. In Proceedings of the 31st International Conference on Very Large Data Bases, 2005.

[14] H Gonzalez, J Han, X Li, and D Klabjan. Warehousing and analyzing massive RFID data sets. In The 22nd International Conference on Data Engineering, 2006.

[15} William A. Florac and Anita D. Carleton, Measuring the Software Process. Addison-Wesley, 1999, ISBN 0-201-60444-2.