Mihailo STANIC, Dragan MITIC, Aleksandar LEBL

Institute for Telecommunications and Electronics, IRITEL A.D. BELGRADE

A Mobile Agents Framework for Integration of Legacy Telecommunications Network Management Systems

Abstract. *This paper presents research on the management integration of legacy network management systems. The proposed integration framework is based on mobile agents, and implements layered decentralized management architecture. In this framework, mobile agents are used to ensure information exchange between legacy network management systems and the top management layer. The roles of these agents, their itineraries and information exchange are described. The framework was implemented and verified in integration of the legacy management system.*

Streszczenie. *Przedstawiono metody zarządzania sieciami telekomunikacyjnymi typu legacy (dziedziczących) bazujące na integrującej strukturze multiagent. (**Struktura mulitagent w zarządzaniu systemami sieci telekomunikacyjnych typu legacy**)*

Keywords: Network Management, Management Architecture, Mobile Agents, Management Integration
Słowa kluczowe: mobile agent, sieci telekomunikacyjne**.**

Introduction

Telecommunications network management systems should provide telecom operators with current information on the network state, with inputs for network planning, and help them to make timely and correct decisions on its provisioning. Equipment from various vendors is typically used in large networks, and various solutions are used for their management. Integration of all of the management solutions is important to ensure that an operator has a uniform set of services for the whole network.

It is still common to find legacy systems in telecommunications networks despite attempts to use open standards. Revolutionary solutions for the management integration of this equipment are usually suggested – it is expected that operators will exchange their existing legacy systems with ones having open standards. This is a costly solution, and in this paper we propose an evolutionary network redesign which preserves the existing infrastructure and saves the operator's existing investments.

The goal is to propose and verify a framework that can be used for the integration of existing legacy systems. It is based upon previous research on service level management in networks using mobile agents [1], and on management architecture [2].

Research on the use of mobile agents in network management is mostly focused on placing them on all entities that form the network, starting with the network elements [3]. This is also the case for solutions for the management of legacy networks [4,5]. A starting assumption in this paper is that minimal changes in the existing infrastructure are required since legacy systems cannot be easily adapted. Therefore, a solution should be in the form of an umbrella over the existing management systems, and any integration should extend their functionality. In this solution, modifications are not required on the network elements or in the protocols for information exchange between network elements and NMS (Network Management System). To adapt the software systems of network elements is an error prone task, and in a network with a large number of elements updating is a difficult and time consuming operation. For legacy NMS's, this would require that modifications are made by the vendor, which would raise additional costs. For each node with NMS there is added an AEE (Agent Execution Environment), this is also added onto nodes for integration and upper management functions.

After this introduction, the layered decentralized management architecture is presented, followed by a description of mobile agent systems and their characteristics. After that, the framework proposal is presented, with a detailed discussion of mobile agent types. Finally, the framework implementation and its verification is described.

Layered Decentralized Management Architecture

The concept of management architecture introduces two entities – manager and managed entity (the term "agent" is commonly used [6]; the focus in this paper is on mobile agents, sometimes calling them just agents, therefore the term "managed entity" is used). A manager calls a set of sequential activities on a managed entity.

In literature [6] are described: centralized management architecture (one manager, many managed entities), manager of manager architecture (adding another centralized layer on top of centralized architecture), and decentralized architecture (distributed managers for distributed managed entities). Following comparison of these solutions it is proposed to define a layered decentralized management architecture [2] which improves management integration, as well as scalability, robustness and cost effectiveness (Fig. 1).

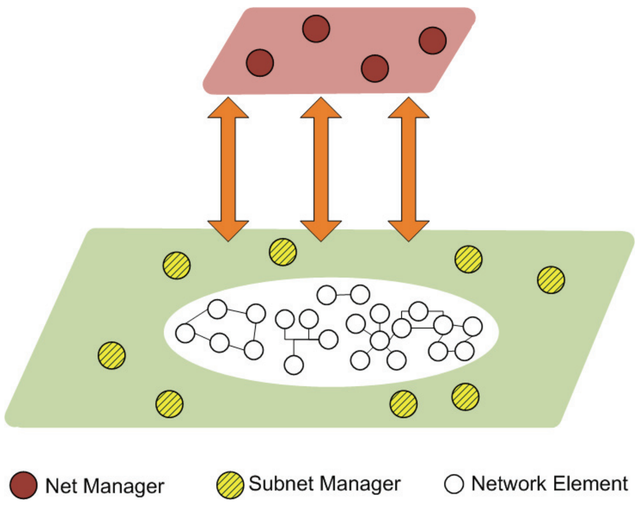

Fig. 1 Model of layered decentralized management architecture

This architecture is intended as a modification of decentralized management architecture, introducing a top management layer into it. The top layer architecture is designed in the form of a distributed system similar to the solution on lower management layers.

Three roles exist in this model: network element, subnet manager and net manager. A group of network elements is managed by the subnet managers. The top management layer are net managers, used to manage subnet managers, and through them network elements. Net managers don't have direct communication with network elements; every operation has to be completed using subnet managers.

This architecture can be created in two steps, in an evolutionary manner – the first step is the introduction of a distributed system on the subnet level, the second step is to add managers on the net level.

Mobile Agents

The proposed management architecture can be described as distributed systems on two layers. Since the nature of systems built on mobile agents is distributed [7], they are chosen as the basis for the framework.

A mobile agent is a computer process that can transport itself on the network from one node to another. They are created on demand, and disposed of when no longer required. Mobile agents can execute only in the AEE, and therefore AEE has to exist on every node where mobile agents need to execute. The mobility of an agent requires transporting its state and class code from one AEE to another, where it resumes execution.

The life of an agent is independent of the process that created it. The agent is deployed to monitor certain system characteristics on one node, executes independently of other agents on that node, but can exchange messages with them. The agent can also wait for certain information to become available, and then process it.

Agents can have support for various protocols used with other system entities, but the communication between them uses protocols defined in the AEE. Some AEEs support message exchange between agents on different nodes; others require that they must be on the same node (local interaction). The second approach is the more common, and it is used in this framework. Since data processing is closer to its source, network traffic is decreased, and the performance of the whole system is better.

Using the distributed nature of mobile agents, one operation can be executed on multiple subnet nodes by an agent that is transferred and executed on all desired nodes. Scalability is thus increased because management activities are distributed.

Security is considered the weakest point of mobile agent technology, and in selecting an agent platform, security mechanisms should be considered.

Framework Description

The framework is designed for integration of legacy systems. It is based on a layered decentralized management architecture, and implemented using agents on the net and subnet layers.

In existing management infrastructure one legacy NMS is executed on every subnet node (Fig. 2) for the management of the group of network elements.

The description of the framework is focused on the integration issues: including legacy NMS on subnet nodes in the mobile agent environment, and collecting information from it, creating new services in the network, and controlling the status of all network entities. The general issues of mobile agents systems are not discussed, such as recovery algorithms when nodes or agents stop functioning etc.
Using agents on network elements would require modification of their embedded software and adding AEE on each network element. A problem is that with older network elements this is not possible – some of them are using older middleware, where AEE cannot be installed. In addition, modifying a large number of geographically dispersed elements also requires their scheduled downtime, which results in a long and costly operation. Since mobile agents are not used on network elements, existing information exchange between network elements and NMS is preserved.

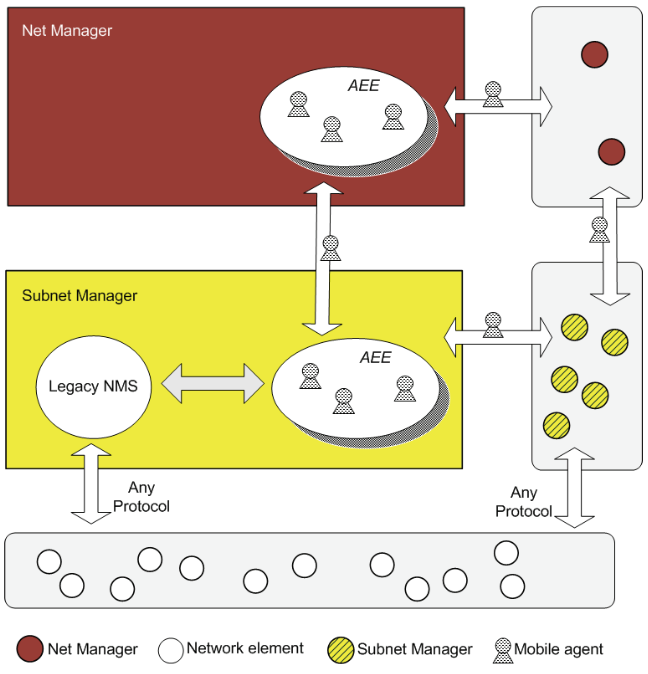

Fig. 2 Model of proposed framework

Integration is provided by adding on to subnet nodes a mobile agent used as wrapper of the existing legacy NMS. The modification of a legacy system is not desirable, since it is closed solution, and should only be done by the NMS vendor. This makes it a costly operation unique for every NMS. Agents are introduced on a subnet layer by adding AEE on each subnet manager. In this solution, NMS is encapsulated by a special agent. Communication to NMS is carried out using its Northbound protocol, this is intended to allow exchanging of information between the NMS and upper management layer.

The net layer is added above the existing legacy infrastructure. Therefore, it can be designed from scratch using mobile agents, with AEE on every net node. The main task of net managers is to initiate integration with subnet managers by sending them various agents used for integration purposes.

Following this analysis, and considering their functionality, three groups of agents were classified: provider, integrator and control agents.

Every agent is presented below by describing its functionality, the way it migrates from node to node, and the number of agents that can exist on one subnet node. UML (Unified Modelling Language) diagrams are used in their description. Agent migration uses UML extension [9].

Provider agents

A provider agent executes on the net manager. Its role is to provide the functions of a net layer, to create integration agents and receive their results. Providers don't have to move from node to node, they are considered static agents. Five different types of providers are distinguished; each agent type is presented below by describing its functionality and the type of integration agents it produces.

The Wrapper Provider is used to create and maintain Wrapper agents that encapsulate NMS functionalities. It ensures that every NMS on a subnet node has its corresponding Wrapper agent. When a Wrapper agent on the subnet node requires updating, this provider sends their new Wrapper agent.

A Control Provider is used to control the state of AEE and residing agents. It creates, sends and receives the results of Control agents' activities. Upon receiving information from returned Control agents, it might initiate framework recovery actions by sending new types of Control agents. This includes creation, disposal, movement, and state change of any other agent in the framework.

A Service Provider provides service management by enabling activities for the establishment of services on the subnet layer on one, or multiple subnet nodes. It creates and sends service agents, and receives their reports of service creation.

An Information Provider is used for collecting information from various subnet managers. It can be used to integrate management functions – fault, configuration, accounting, performance and security. An Information Provider creates and sends Information agents, and receives gathered information as a result of their activities.

An Integration Provider is used for the integration of the framework with other management infrastructures that are not agent based. In this way, an open framework is created, and is not bound to the agent environment.

Integration agents

An Integration agent, created by Integration Provider, travels to one or more subnet managers, and executes on each visited node. Since the result of their activity should be delivered to its corresponding provider agent, after completion of activities they return to the node where the provider resides and sends it the results. Three types of integration agents are proposed: Wrapper, Service and Information agents.

Integration agents – Wrapper agent

A Wrapper agent is used to encapsulate NMS in the agent environment. A legacy NMS on a subnet node doesn't have an interface to the agent environment, but implements some form of Northbound interface.

Each legacy NMS product should have a corresponding Wrapper agent, adapted to its specification. A Wrapper agent extends the functions of a NMS by keeping settings that don't exist in the NMS. This enables the easy addition of new features, such as introducing sub-networks that can be managed from various NMS, introducing services, additional processing of NMS data (collecting predefined set of alarms etc.)

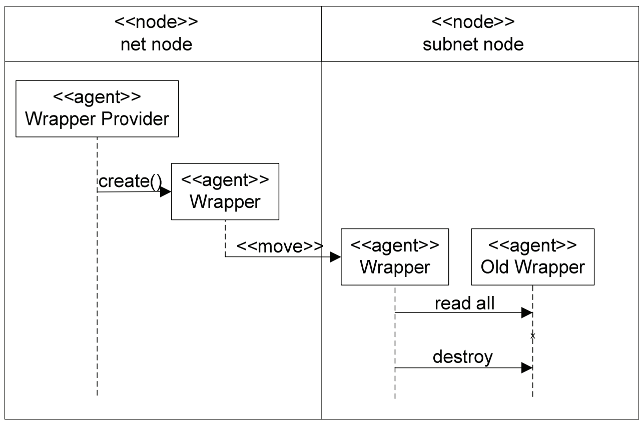

Fig. 3. Update of Wrapper agent

Figure 3 shows the scenario of creating a new Wrapper agent for a selected subnet node. After being sent by the Wrapper Provider from the net manager, the Wrapper agent is transferred to the corresponding subnet node, where it stays while it executes. Data is processed locally, which is the fastest and most reliable way.

When a Wrapper agent arrives on a selected node, it iterates through the present agents, checking if there already exists another Wrapper agent. If it does – an update is needed, and the new Wrapper agent transfers all of the information set from the old agent, then initiates its disposal. In this way, when a Wrapper agent is updated, any previously collected information is preserved.

Exactly one Wrapper agent exists on each subnet node.
Integration agents – Service agent

A Service agent is used for establishing services in the network. The service is created during the exchange of information with a Wrapper agent.

A Service agent is initiated by a Service Provider, and then sent to subnet managers. When the establishment of a service requires setting parameters on multiple NMS, the Service agent travels to multiple nodes, and contacts multiple Wrapper agents. After performing the desired behavior, they return to Service Providers, deliver the results of the successful service creation, and disposes of itself. Figure 4 shows creation of services on subnet node 1 and node 2 using a service agent.

An arbitrary number of Service agents can exist on each subnet node.

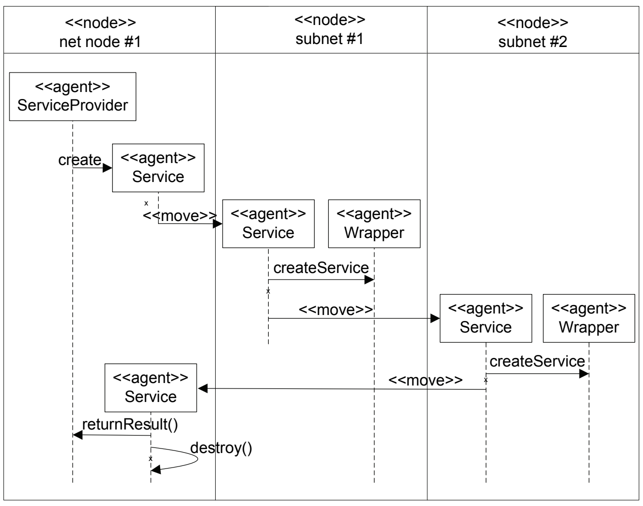

Fig. 4. Service agent execution

Integration agents – Information agent

An Information agent is used for the transfer of information from net to subnet managers, and vice versa. It might be used to implement some of the management functions – fault, configuration, accounting, performances and security. On a target subnet node it exchanges information with Wrapper agent.

Information agent goes from Information Provider on net manager layer to subnet managers. If information exchange with multiple NMS is required, it is transferred between multiple subnet nodes. After finishing activities, it returns to node where it was created, delivers the resulting data to its parent Information Provider agent, and disposes of itself. A scenario for sending an Information agent to two nodes is similar to that of service agent, depicted in Fig. 4.

An arbitrary number of Information agents might exist on a subnet node.

Control agents

A control agent is used for monitoring network resources and agents in the framework, and sending this information to a Control Provider. Considering network resources, this includes control of nodes on net and subnet layers, and the operational state of their corresponding AEE. It also includes discovery of agents on selected nodes, and a check of their performance. Different types of Control agents collect information on active agents in the framework, discover new resources in network, initiate actions for load balancing of traffic, conflict resolution, etc.

Various itineraries are used for the transfer of Control agents, and one of them defines circulation on all network nodes. The number of Control agents on each net and subnet node is arbitrary.

Framework Implementation

Implementation is done using Aglets Mobile Platform [9]. This is a general purpose Java-based mobile agent platform, which enables agent migration using Agent Transfer Protocol. Communication between agents is provided using messaging. Embedded security prevents malicious agents from accessing prohibited data. The classes of mobile agents must be derived from the abstract Aglet class (since this is the case with all the mobile agent classes, the Aglet class is not presented on class diagrams in this paper).

The proposed framework was tested in the integration of multiple subnet nodes with SUNCE-M [10], and a legacy NMS designed by IRITEL. SUNCE-M is used for continuous network monitoring of telecommunications transmission systems made by IRITEL. It implements element and network management layers, and supports the following management functions: fault, configuration, performance, and security. A proprietary protocol is used for data exchange with network elements; it also provides a Northbound interface in the form of XML files, delivered using a publish-subscribe model.

For testing purposes there were used 5 subnet nodes with legacy SUNCE-M. On the top layer were 2 net nodes.

The itineraries for agents are defined using abstract class Itinerary, and its derived classes – *OneNodeItinerary* defines migration to one node, *MultiNodeItinerary* defines migration on multiple nodes

Provider agents described in the framework description were created, using classes: *WrapperProvider*, *ControlProvider*, *ServiceProvider*, *InformationProvider* and *IntegrationProvider*. The last class was created only for verification purposes, while the rest of them were used to create integration or control agents and send them on the required itineraries. In this framework implementation, exactly one agent of each provider class was active. An improvement for high availability can be achieved by adding redundancy, for example by having active one provider per node of the net layer. This would require algorithms that would keep the same information base on all providers, and setting a provider as active or passive would be needed when it creates new integration or control agents.

In this implementation, each type of integration and control agent is represented by an abstract class.

The functions of the abstract Wrapper agent (class diagram presented on Fig. 5) include its transfer to a selected node described using *OneNodeItinerary* class, and its installation and execution on that node. It provides communication with other agents using adequate message types. A derived class *NMSWrapper* was designed, with implementation details of the legacy NMS. Its information model and their retrieval are encapsulated in class *XMLbridge*.

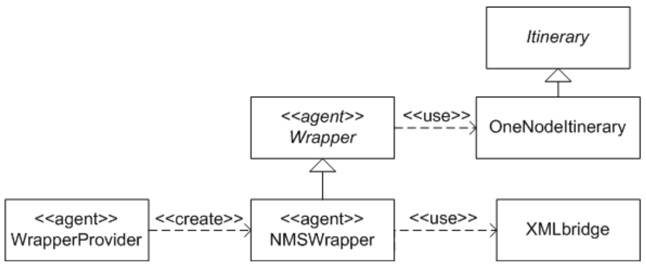

Fig. 5. Class diagram of Wrapper agent

The abstract class *Integrate* is the base for abstract classes *Service* and *Information* (Fig. 6). It provides agent migration to multiple nodes using the *MultiItinerary* class. It also implements a function for finding a Wrapper agent on the subnet node. Two services are described in derived classes *ServiceA* and *ServiceB*. The format of the expected result is also defined in those classes. The process of creating new services starts when *ServiceProvider* creates one of the derived service classes, and initiates its service parameters (for example devices and ports) using the *ParamDescription* class.

Similar is the use of the *Information* class – its derived class *InfoAlarm* is created using *InfoProvider* class, and its parameters are initiated using *ParameterDescription* class. Information exchange with a *Wrapper* agent and returning results to *InformationProvider* are defined in the derived *InfoAlarm* class.

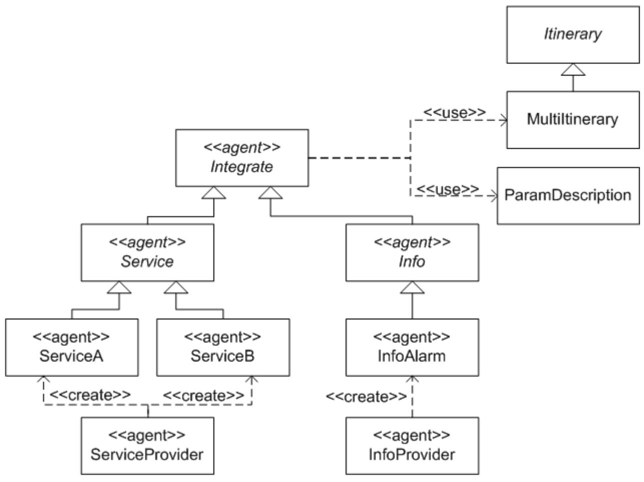

Fig. 6. Class diagram of Service and Info agents

Using late bound characteristics of Java, adding new service in this implementation is equivalent to creating a new derived Service class, and adding it into agent repositories. The same mechanism is used for a new information exchange scheme – it is only required to add the new derived Information classes into the class repository.

The abstract class *Control* (Fig. 7) is the base class that declares control functions and supports the migration plan using *MultiItinerary* for migration on multiple nodes. The agent itinerary is initiated in the creation of the agent by *ControlProvider*. In this implementation, *ControlCirculate* is derived class of *Control.* The agent of this class travels across all the nodes in network. It is used to discover all agents present on one node and to check their state.. When the agent returns to the node of origin, it sends its results to *ControlProvider*.

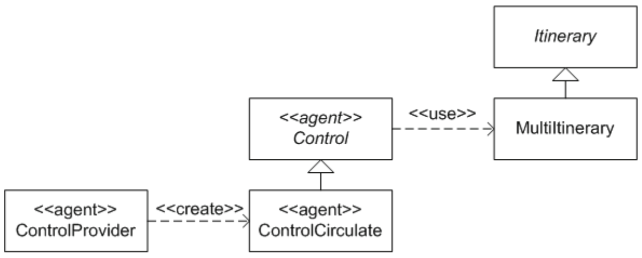

Fig. 7. Class diagram of Control agent

Conclusion

This paper proposes a solution for the management integration of the legacy systems where adaptation of network elements is not a feasible solution. The proposed framework can be used for management integration of any NMS with Northbound interface. The work of other authors using management integration with mobile agents starting from network elements is discussed. This method has an implicit assumption that adaptations can be made on network elements, in practice this is not always possible. This problem becomes complex when using proprietary management solutions.

The proposed solution is deployed on nodes with NMS and nodes on the top management layer. Instead of creating a unified protocol between various NMS and network elements, existing protocols are preserved. Also, this solution can be used when vendor agreement on the modification of existing management solutions is hard to negotiate. By reusing existing management solutions (NMS and software on network elements) the cost of integration is decreased, and its deployment time is shorter.

The proposed framework is based on a layered decentralized management architecture, which can be described as interconnected layered distributed systems. Since mobile agents are established as distributed systems, they are chosen for the implementation.

In the framework, agents are classified into three groups: provider agents, used on the top management layer, control agents used on all management stations for the control of complete framework, and integration agents used to process information on nodes with NMS running and its transfer to the top management layer.

The application of the solution is demonstrated in the framework implementation with Aglets Mobile Platform used as the toolkit for the mobile agents implementation. The integration uses a legacy system designed by IRITEL – SUNCE-M. During the implementation of the framework different scenarios were verified: installing agents used as wrappers for legacy NMS and updating them, creating new services, transferring information between layers, and controlling the nodes and agents execution.

It is demonstrated that the proposed framework can be used for integration of multiple legacy NMS used in telecommunications transport network. Further integration of a new legacy NMS in the implemented framework would require the creation of classes that encapsulate both the information model and Northbound interface of that NMS.

ACKNOWLEDGEMENT

This work is partially supported by the Ministry of Science and Technological Development of the Republic of Serbia under Grant TR32007.

REFERENCES

[1] Stanić M., An integration on upper layers of network management based on mobile agents, *ETRAN 2003, Conference Proceedings*, Vol. 2., pp 135-138, 2003.
[2] Stanić M., Mićić N., Vulićević V., Knežević P., Ilić M., Katanić D., A Layered Decentralized Management Architecture in Integration of SUNCE-M with SUNCE+, *Proceedings of IEEE TELSIKS 2011*, Vol.2., pp. 729-732., Oct. 2011.
[3] Gavalas D., Tsekouras G. E., Anagnostopoulos C., A mobile agent platform for distributed network and systems management, *Journal of Systems and Software*, Vol.82, No.2, pp. 355–371, Feb. 2009.
[4] Simoes P., Reis R., Silva M.L., Boavida F., Enabling mobile agent technology for legacy network management frameworks, *Proceedings of IEEE Softcomm '99*, Oct. 1999.
[5] Cardoso A., Celestino JR.J., Celestino R., Management of Heterogeneous ATM Networks Based on Integration of Mobile Agents with Legacy Systems, *In Proceedings of the Network Operations and Management Symposium (NOMS 2002)*, Florence, Italy, April 15th – 19th, 2002.
[6] Chen G., Kong Q., Integrated Telecommunications Management Solutions, *IEEE Press*, 2000.
[7] Lange B.D., Oshima M., Seven good reasons for mobile agents, *Communications of the ACM*, Vol.42. No.3, pp.88-89, March 1999.
[8] Aglets Mobile Agent Platform, IBM, http://www.trl.ibm.com/aglets
[9] Kusek M., Jezic G., Modeling Agent Mobility with UML Sequence Diagram, *Agentlink III – AOSE TFG2*, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. Online: http://www.pa.icar.cnr.it/cossentino/al3tf2/docs/ag_mobility.pdf
[10] Stanić M., Katanić D., Knežević P., Vulićević V., Software Architecture Of Management System For SDH And PDH Telecommunications Equipment – SUNCE-M, *ETRAN 2002, Conference Proceedings*, Vol.44, No.12, pp. 2395-2503.

***Authors**: mr Mihailo Stanić dipl.ing., IRITEL A.D., Batajnički put 23, 11080 Belgrade, Serbia, phone 381-11-3073456; e-mail: mihailo@iritel.com; dr Dragan Mitić dipl.ing., IRITEL A.D., Batajnički put 23, 11080 Belgrade, Serbia, phone 381-11-3073425; e-mail: mita@iritel.com; dr Aleksandar Lebl dipl.ing., IRITEL A.D., Batajnički put 23, 11080 Belgrade, Serbia, (phone 381-11-3073422; e-mail: lebl@iritel.com*