A transformation formalism to model the Management Information lifecycle

Livia Vaculova[1], Daniel Ranc[1], Siham Elmejjad[1]

[1] Institut National des Télécommunications, Network and Software Department,
9 Rue Charles Fourier, 91101 Evry, France

{Livia.Vaculova, Daniel.Ranc,
Siham.Elmejjad}@int-evry.fr

Abstract. This paper describes an operator workflow oriented information framework for services and network management. Particular attention is drawn on the requirement for a high-level management information specification. It proposes for this purpose a management information transformation formalism as a tool to specify the polymorph behavior of the information during its lifecycle across the operator workflow. Actually, the resulting management entity called Management Applet (Manlet) features intelligence, polymorphism and mobility in order to perform sophisticated management tasks at the service and customer level.

Keywords – Framework Architecture, Information Model, Information Lifecycle, Operators, Polymorphism, QoS, Scenario, Service & Network Management, SLA, SLS, Transformation Formalism, UML.

1 Introduction

This paper is a transient step of an industrial research project which aim is to elaborate a new Telco workflow oriented information framework. A first step was focused on the definition of a new flexible, powerful, end-user oriented information model featuring the high-level abstraction that is required for network and service management. Starting from the analysis of realistic operator business cases and workflow analysis, the requirements for information modeling have been deduced and, using the UML approach and tools a complete information model has been specified [8]. A suitable framework architecture has been developed attempting to cover telecom stakeholder needs that have never been dealt with seriously in other approaches.

The fast evolution of the Telco market in terms of new fast, real-time, low jitter and multimedia services brings the need to change the existing point of view on management information. The emerging landscape of new network technologies such as Gigabit Ethernet, WDM, radio LAN etc. puts into a new light the requirements for transparent, technology independent management information.

This information is for Telcos of crucial importance and tends to be more intelligent, autonomous, mobile, relevant for efficient service and network management as well as oriented to its end user needs. This implies several transformations during the lifecycle of the management entity, which increases its complexity. The information is born, lives and finally dies. It changes its behavior, its status and adapts itself to the different situations it meets.

The conceptual resources addressing seriously the question of polymorph specification are scarce. It is indeed quite surprising that state of the art specification tools and methods tackling sophisticated matters such as entity-relationship models, scenario-based approaches etc., lack of any suitable formalism to model polymorph behavior of objects. However the Telcos for which this formalism is destined put usually strong requirements on formal tools with a clean and complete design. A radically new information transformation formalism is therefore needed. This paper focuses exactly on this question, proposing a formal approach to model Management Information polymorphism.

The paper is structured as follows. First, section 2 recalls the fundamentals of the project and the results of the previous achieved tasks. Section 3 is devoted to the description of the information transformation formalism. The formalism implementation is shown on the scenarios in the last section.

2. Information Framework for Services and Network Management

The project was motivated by the strong limitations of the existing management frameworks that have been driven purely by technical motivations and did not cover any service or Telco business related aspects.

Each management system is composed of three entities: the Protocol, the Application and the Information. Existing systems rely either on the Application (mainstream approach of centralized computing), other are strongly supported by the Protocol (case of management systems such as TMN, where CMISE/P takes in charge many powerful functions). One of the original contributions of this project is the choice to put the emphasis on the Information in order to make it more intelligent and more autonomous to be able to address functions that are usually insured by the Protocol or the Application in existing systems.

Our lifecycle oriented information framework defines the complete information model and framework architecture for service and network management [5][6]. The model was developed using the Unified Modeling Language (UML) [7] that is a widely recognized standard modeling method for complex and abstract systems design. We must suggest that despite its power, UML does not provide support to model polymorph behavior.

2.1 Static Management Information Model

Telco activities have been deeply analyzed in order to extract the fundamental scenarios that enable us to make an initial project specification suitable to the requirements for a network and service management information framework. A dynamic study along with a static one allowed us to define the information model composed of the most representative classes and the relations between them. The static model of the system is divided into two categories:

2.2.1 Customer Management

Figure 1 represents the customer management class diagram of our information model.

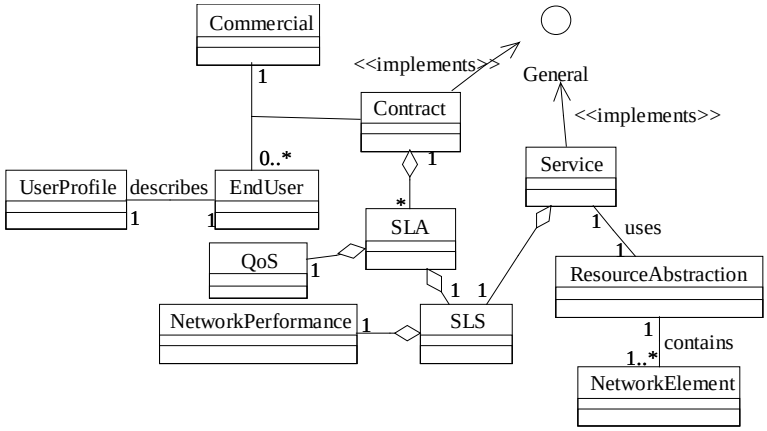

Fig. 1. Customer Management classes diagram

The *UserProfile* class describes the *EndUser*'s preferences and habits. The *EndUser* class is associated to the *Commercial* class through an association class *Contract* that is composed of one or more *SLA* classes. The *SLA* class contains the *QoS* class that describes the *QoS* parameters [2]. To provide the end-to-end *QoS*, the *SLA* is mapped to one *SLS* class containing corresponding technical characteristics [1]. The *ResourceAbstraction* class is an abstraction of the set of *NetworkElements* responsible for a service delivery.

2.2.2 Network Management

Figure 2 models the network management class diagram of our information model. The *Technician* maintains the network devices. The *NetworkElement* instance can trigger zero or more *Alarms*. The trouble that triggered the *Alarm* can impact *Services* and *EndUsers*. The *ImpactedEndUser* and *ImpactedService* association classes between the classes *Alarm*, *Service* and *EndUser* allows the *Alarm* to be enriched during its life cycle and enables the polymorph vision of the management information or ManLet.

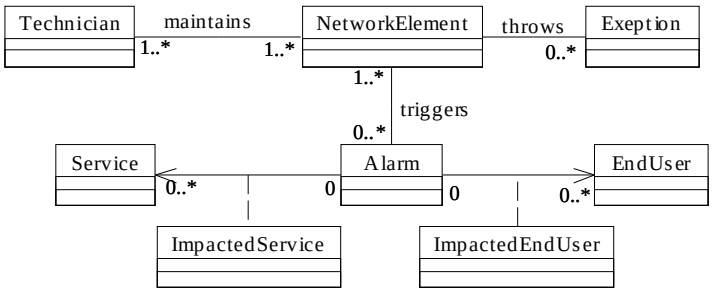

Fig. 2. Network Management class diagram

2.2 Framework Architecture Summary

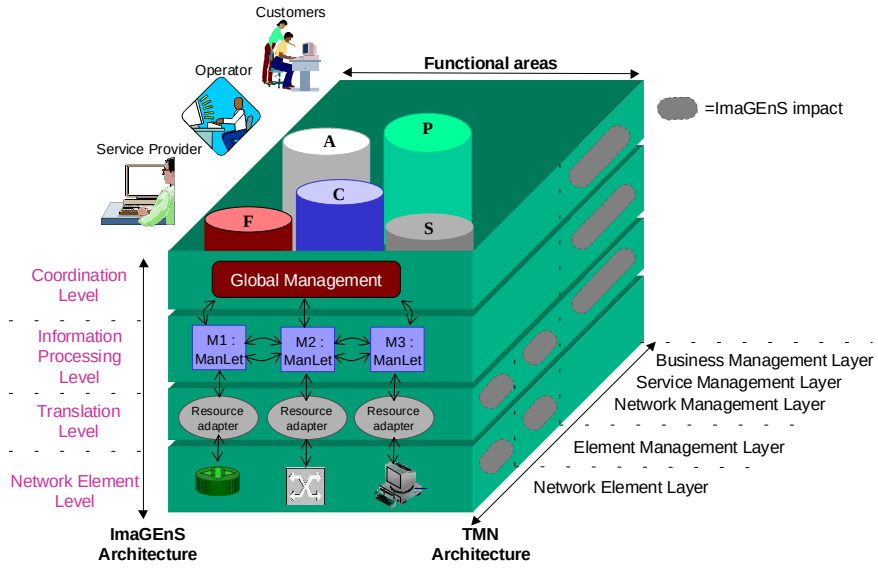

Fig. 3. Framework architecture

The framework architecture (Figure 3) shows how the different entities of the system cooperate. The network equipments are represented in the first level, and are linked to the system by means of the generic resource adaptor placed in the Translation Level. The resource adaptors take in charge the translation between low-level information (SNMP traps, SNMP commands, IOS command line) and the ManLet residing on the Information Processing Level. The composition and combination of all the local ManLet behaviors results in global management behavior at the Coordination Level.

3. High Level Information Transformation Formalism

Nowadays existing transformation formalism methods do not encompass the scope of information transformation during its lifecycle. These legacy methods are based on abstract mathematic expressions or algorithmic languages and aim at optimizing huge programs and process execution.

The transformation of the information responsible for the network and service management in our system is formalized through a number of single or macro-operators. This formalism represents a generic model of the way the information transforms itself during its lifecycle. It is a high-level abstraction of this transformation so that it can be applied to any information in any situation.

It should be kept in mind that this formalism describes the information transformation lifecycle from a conceptual point of view, which is not the implementation in any programming language. The latter pertains to a further step of the system validation and implementation.

3.1 Transformation Operators

We analyzed states in which our information exists through different ManLet lifecycle scenarios presented in the last section. This deep study led us to specify information state transformation operators shown on Figure 4 the ManLet object uses to afford its polymorph and autonomy characteristics.

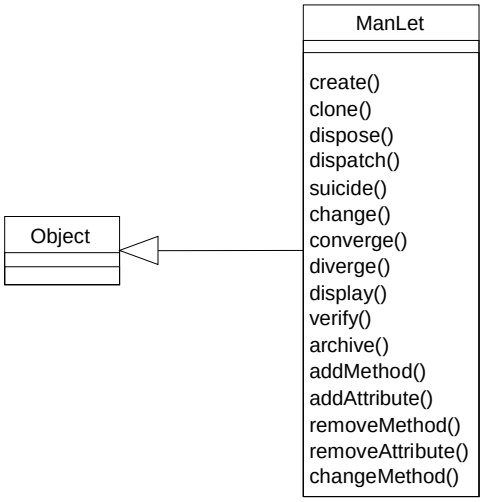

Fig. 4. The ManLet is an object that uses the transformation operators to change its state.

– `ManLet create()`

This operation gives birth to the management information.

– `dispose(M:ManLet)`

Disposes the ManLet M passed as a parameter.

– `suicide(M:ManLet)`

Allows the ManLet M to destroy itself at the end of its lifecycle.

– `ManLet clone(M:ManLet)`

The original ManLet M is duplicated and a new cloned ManLet object with new specific identifier is returned.

– `dispatch(M:ManLet,D:InetAddress)`

The ManLet M is sent to the destination address D.

– `display(M:ManLet)`

The information carried by the ManLet object M is displayed accordingly to the targeted staff e.g. technical or commercial.

– `ManLet[] diverge(M:ManLet)`

The operator splits ManLet M into many others ManLet objects.

– `ManLet converge((M1,M2,…,MN):ManLet[])`

This operator combines a set of ManLet objects (M1,M2,…,Mn) in a unique one.

– `boolean verify(M:ManLet)`

This operator verifies the validity of the user request e.g. service cancellation. It compares the attribute values of the ManLet object in the argument to the corresponding object attributes values stored in a database. If the compared attribute values equals the operator returns True, False otherwise.

– `boolean archive(M:ManLet,DB:Database,T:Timestamp)`

This operator archives the concerned ManLet object information in a database DB. The parameter T represents the information validity duration after which the information will be destroyed. It returns true if the archiving is achieved successfully, false otherwise.

– `ManLet change(M:ManLet)`

This macro-operator gathers a set of operators that are used sequentially to make the desired change on the ManLet M transmitted as a parameter. It returns the changed ManLet object.

The properties describing the ManLet behavior and characteristics during its lifecycle are added or removed according to the state of the information thanks to following operators:

– `ManLet addAttribute(M:ManLet,AttName:String,AttValue:Object)`

Adds the attribute called AttrName with its specific value AttValue to the ManLet M. It returns the ManLet resulting from the attribute addition.

– `ManLet removeAttribute(M:ManLet,AttName:String,AttValue:Object)`

The attribute called AttrName having the value AttValue is removed from the ManLet M. A ManLet resulting from the attribute retrieval is returned.

– `ManLet addMethod(M:ManLet,MethodName:String)`

Adds the method called MethodName to the ManLet M. It returns the ManLet resulting from this addition.

– `ManLet removeMethod(M:ManLet,MethodName:String)`

The method called MethodName is removed from the ManLet M. A ManLet object resulting from the method retrieval is returned.

– `changeMethod(MethodName:String)`

The method called MethodName is changed in order to perform the intelligence of the ManLet that will be able to change dynamically its behavior.

4 Formalism Implementation Scenarios

Following "Pre-Conditions" are required to launch our scenarios:

1. The telco provides the access to the end-user personal web page in the telco's web-site. This web page contains the information about the end-user services subscription state. Through this medium, the customer is allowed to request, add or make changes on the services.
2. The end-user has already an established contract containing a list of services it has subscribed to.
3. The end-user is authenticated before having access to its personal web page.
4. We focus on SLA established between a customer and a Telco because it needs more complex analysis e.g. SLA to SLS translation.
5. We consider the Telco able to provide the services he suggests to its customers anywhere and at anytime with the specified QoS they have agreed.

4.2 Add Service Scenario

Figure 5 illustrates the ManLet states transformation. The management information acquires new properties or looses some of them during its lifecycle, either attributes or methods.

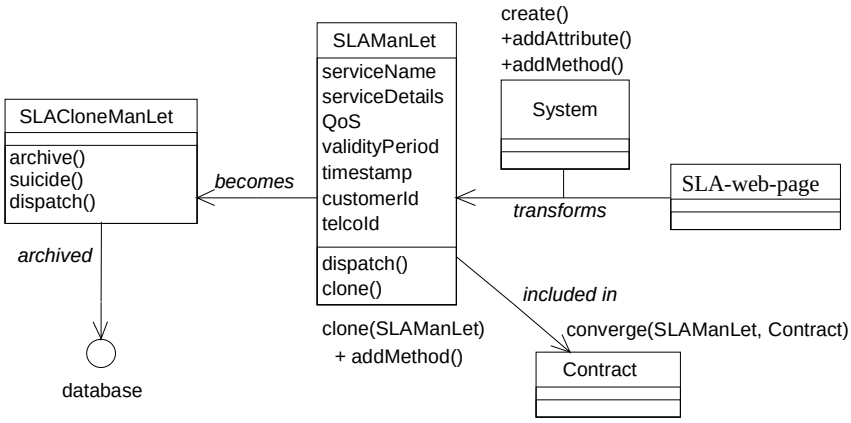

Fig. 5. Instance of the state-chart diagram of add service scenario. It gives a detailed view of the information transformation e.g. added/removed properties

This scenario begins when the end-user expresses its will to add a new service, e.g. videoconferencing with a high QoS, to its contract. The Telco suggests a set of services to the end-user with theirs description containing service specific parameters and service delivery requirements predefined by the Telco. The customer specifies certain aspects of the service he intends to purchase, such as the date and time the service usage will begin and

length of time it will be used for. Upon receipt of the acceptance of both, the customer and the Telco, an *SLA_web-page* script will be sent to the *System*.

The *System* transforms the latter to the first ManLet *SLAManLet* using some single or macro-operators as follows:

```
SLAManLet=create()
addAttribute(SLAManLet,ServiceName,Videoconf)
addAttribute(SLAManLet,ServiceDetails,url1)
addAttribute(SLAManLet,QoS,High)
addAttribute(SLAManLet,ValidityPeriod,(01/01/03,08:00,12/01/03,08:00)
addAttribute(SLAManLet,Timestamp,31/12/02)
addAttribute(SLAManLet,CustomerId,Jean Jolie)
addAttribute(SLAManLet,TelcoId,Ztelecom)
addMethod(SLAManLet,dispatch())
addMethod(SLAManLet,clone())
addMethod(SLAManLet,archive())
```

The end-user has already a *Contract* with the Telco composed of SLAs. Using the operator *converge* a new SLA is included in the contract.

```
Contract=converge((SLAManLet,Contract))
```

The *SLAManLet* becomes *SLAManLetClone* executing the clone operator in order to be archived into the Telco private distributed database and in the public user profile database shared between all the Telcos to finalize its lifecycle.

```
SLAManLetClone=clone(SLAManLet)
archive(SLAManLetClone,telcoPrivateDatabase)
archive(SLAManLetClone,telcoPublicDatabase)
```

4.3 Service Request and Delivery Scenarios

The customer requests the VoIP service delivery. This request is sent to the *System* that will look for the corresponding *SLAManLet* that is archived in the database and translate it into a *SLSManLet* containing technical specifications [3] [4]. This information transformation is modeled on Figure 6.

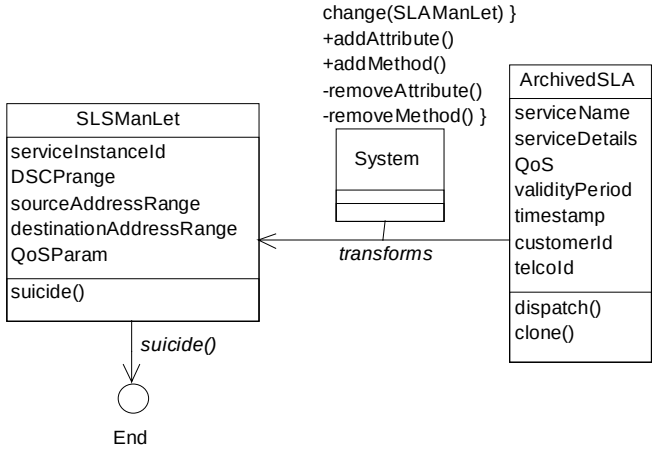

Fig. 6. Instance of the state-chart diagram of service request and delivery scenarios, giving a detailed view of the information transformation e.g. added/removed properties

To provide this translation we need to add and remove some attributes of the management information. This leads us to define the macro-operator change that in this case will be composed as follows:

```
ManLet change(M:ManLet){
addAttribute(SLAManLet,ServiceInstanceId,1AB2)
```

```
removeAttribute(SLAManLet,ServiceDetails,url)
removeAttribute(SLAManLet,QoS,High)
addAttribute(SLAManLet,DSCPrange,24)
addAttribute(SLAManLet,SourceAddressRange,(157.159.100.245,255.255.255.0))
addAttribute(SLAManLet,destinationAddressRange,(157.159.100.246,255.255.255.0))
addAttribute(SLAManLet,QoSParam,(10,125,10^-3,25000))
}
SLSManLet=change(SLAManLet)
```

The technical information performed by the *SLSManLet* allows the generic Resource Adaptor to take in charge the resource configuration.

Once the service is delivered to the customer, the *SLSManLet* suicides itself in order to avoid database overload.

```
suicide(SLSManLet)
```

4.5 Alarm Propagation Scenario

Figure 7 demonstrates the evolution of the ManLet information transformations during this scenario.

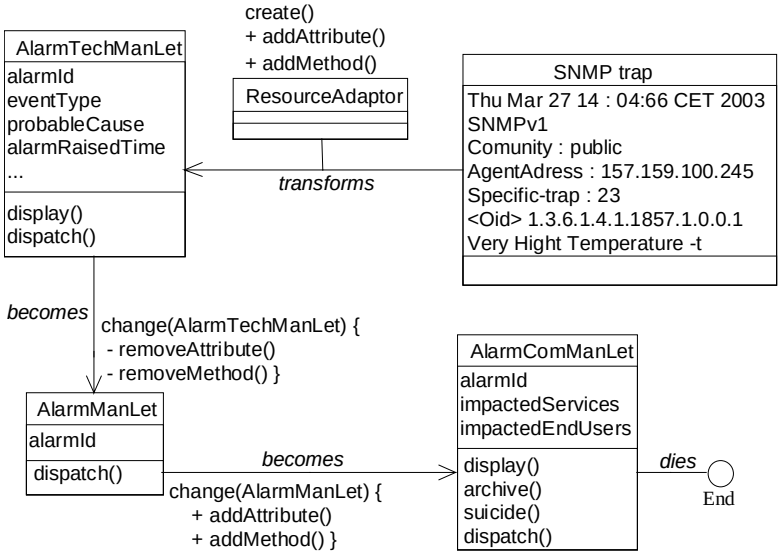

Fig. 7. Instance of the state-chart diagram of alarm propagation scenario. It gives a detailed view of the information transformation e.g. added/removed properties

In case of trouble on the network, the concerned network element triggers an alarm in the form of a *SNMP trap* containing the purely technical equipment-based information.

The *ResourceAdaptor* captures this low-level information and creates a new *AlarmTechManLet* to which the attributes and methods are added accordingly to the management information model[8] described in the section 2.

```
AlarmTechManLet=create()
addAttribute(AlarmTechManLet,AlarmId,1)
addAttribute(AlarmTechManLet,eventType,communication error)
addAttribute(AlarmTechManLet,ProbableCause,Linkdown)
addAttribute(AlarmTechManLet,AlarmRaisedTime,20.03.03 10:26)
addMethod(AlarmTechManLet,dispatch())
addMethod(AlarmTechManLet,display())
```

The *AlarmTechManLet* is then dispatched to its first destination, e.g. the technical staff and it is displayed.

```
dispatch(AlarmTechManLet,157.159.100.246)
display(AlarmTechManLet)
```

Before being dispatched to the database to be enriched for commercial purposes, the *AlarmTechManLet* is changed in order to clarify the information for the commercial staff. In this case the operator change is defined as follows:

```
AlarmManLet=change(AlarmTechManLet){
removeAttribute(AlarmTechManLet,eventType,communication error)
removeAttribute(AlarmTechManLet,ProbableCause,Linkdown)
removeAttribute(AlarmTechManLet,AlarmRaisedTime,20.03.03 10:26)
removeMethod(AlarmTechManLet,display())
}
dispatch(AlarmManLet,database_url)
```

The *AlarmManLet* will be enriched in the databases by the impacted end users and services information. The *AlarmManLet* status changes into the *AlarmComManLet* status through the change operator:

```
AlarmComManLet=change(AlarmManLet){
addAttribute(AlarmManLet,impactedServices,[VoIP,VoD])
addAttribute(AlarmManLet,impactedendUsers,[Stefan,Azzeddine])
addMethod(AlarmManLet,archive())
addMethod(AlarmManLet,display())
addMethod(AlarmManLet,suicide())
}
```

The lifecycle of the *AlarmComManLet* continues. It is dispatched and displayed. The commercial staff can take in charge the customer relationships management.

```
dispatch(AlarmComManLet,157.159.100.245)
```

```
display(AlarmComManLet)
```

The lifecycle of the *AlarmComManLet* is ended. Finally, it is archived if necessary and then suicides itself.

```
archive(AlarmComManLet,AlarmListDatabase)
suicide(AlarmComManLet)
```

5 Conclusion

This contribution has been the building ground for a completely new and original ManLet transformation analysis. The goal has been the definition of a radically new high-level information transformation formalism describing the information lifecycle.

This formalism attempts to tackle seriously the management information polymorph behavior. It details the exact way the information transforms itself through the different states. This formalism will be used as a central tool to specify further work, in particular the demonstration scenarios of the study.

Future reflections may propose the implementation of an interpreter of the proposed transformation language e.g. a system able to read formal transformation specifications and to generate automatically from them new landscapes of live ManLets.

References

1. Dugeon, O., Diaconescu, A.: From SLA to SLS up to QoS Control: the CADENUS Framework WTC'2002
2. TeleManagementForum: SLA Management Handbook, GB917 June 2001
3. Inter-operator interfaces for ensuring end-to-end IP QoS, Project P1008, Deliverable 2 volume 1 & 2, Deliverable 3 part 3, 2001.
4. Cisco systems, Service Level Manager Programmer's Guide, 2000-2001
5. ITU-T Rec.: M3100: Generic Network Information Model, October 1992
6. TINAC: Definition of Service Architecture version 5.0 1997
7. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Westley Reading MA 1999
8. Vaculova, L., Ranc, D., Elmejjad, S.: A Life Cycle Oriented Information Framework to Manage Networks and Services, ConTEL 2003