A method of automatic assessment of feature compatibility in mobile networks

Szymon Fedor and Liam Fallon
Network Management Lab
Ericsson Software Campus
Athlone, Ireland
Email: {szymon.fedor, liam.fallon}@ericsson.com

***Abstract*—Deployment and upgrade of a mobile network have always been challenging tasks. Very often they require human intervention because telecom networks are complex systems composed of different nodes that need to be compatible in order to communicate and provide network services. Therefore in current telecommunication systems a network expert must check all the requirements and compatibilities of the network prior to activation of a new service. Automation of the assessment of network compatibility is one of the key enablers for Autonomic Management of telecom networks. In this paper we describe a new method for automatic end-to-end assessment of compatibility between network features in a telecom network. The method enables fast, easy and accurate decision making regarding the planning of new feature deployment or the upgrade of already existing features. We built a prototype that demonstrates the described method. It shows that our method is not bound to any type of telecom network and could be used to automate deployment or upgrade of a multiple-domain network.**

***Keywords*-Autonomic Management; feature compatibility; mobile network;**

I. Introduction

The operation of a mobile network is enabled by network features which work together to give a correctly functioning telecom system. An operator must deploy basic features which are the minimum requirements for management of a network, e.g. SMS over GPRS in HLR[1] or Emergency Call[2]. In addition to that a network operator can decide to install enhanced or optional features (e.g. SGSN In Pool[3] or HSDPA Mobility[4]) in order to improve network capabilities.

Telecommunication networks, and the features offered on those networks, are becoming larger and more complex. It is very commonplace to see network features implemented in many network elements which are often located in different network domains. For example, the "SGSN in Pool" feature is implemented in SGSNs in the mobile core network and in RNCs and BSCs in Radio Access Networks.

Network features are increasingly implemented largely or solely in software. The concept of software defined radio heralds an era where all radio and digital signalling processing is defined using software. Software embedded in network elements uses enabling technology such as web services and peer to peer techniques to implement advanced Self Organising Network (SON) features such as Automatic Neighbour Relations (ANR)[5].

Today, the software for most network elements is bundled into a single load module. All the software programs needed to implement the features supported by a network element (which might include part of network features) are packaged into this single load module. Ensuring that all network features are compatible across all nodes that implement features is currently handled as an off-line administrative activity.

The current process of software installation and upgrade in a telecom network is inefficient especially in the future context when software packages will be deployed more frequently then nowadays. An upgrade or deployment of a new feature is preceeded by a careful back office study to ensure compliance among all network components after activation of that new feature. This analysis of feature compatibility is usually performed manually and therefore is error-prone and expensive because it requires expertise in various telecom domains.

In this paper we describe a new method for an automatic assessment of feature compatibility prior to upgrade or deployment of a new feature in a network. The proposed solution will reduce the cost and effort related to the planning and deployment of new network features. It will enable the introduction of an automatic process for checking

[1]The SMS over GPRS in HLR feature provides a subscriber with the possiblity to send and receive Mobile Terminated and Mobile Originated Short Message Services (SMS) over the General Packet Radio Service (GPRS) in a GSM network.

[2]In WCDMA network Emergency Call feature supports emergency calls within WCDMA RAN. It also enables emergency call setup in congested WCDMA cells where normal voice calls would be rejected.

[3]In WCDMA network SGSN In Pool makes it possible to connect an RNC to several SGSNs.

[4]HSDPA Mobility enables the HSDPA user to change cell with a minimum of user data interruption.

[5]ANR allows new cells or cell clusters to automatically populate their neighbor lists and the reciprocate cells neighbor list.

978-1-4244-6557-6/10/$26.00

feature compatibility during the network planning or upgrade phase. It could be linked to a future licensing system, where a user can automatically estimate required software packages to activate a network feature and then request a license for them. Section II provides an overview of solutions to the feature compatibility problem from domains other than telecom. In section III we explain why it is important to solve the addressed problem from current and future perspectives. Then, sections IV and V describe an overview and details of our new method that automates the assessment of feature compatibility in telecom networks respectively. We present the prototype of a tool that supports the new method in section VI. Finally, in section VII we conclude the paper.

II. Related work

The problem of feature compatibility has been identified in several domains outside telecommunications. Research to solve this problem has been ongoing for a long time. We have identified three main domains into which the state of the art can be assigned.

A. Software upgrade in distributed systems

Upgrading the software of long-lived, highly-available distributed systems is difficult. It is not possible to upgrade all the nodes in a system at once, since some nodes may be inaccessible and halting the system for an upgrade is unacceptable. This means that different nodes may be running different software versions and yet need to communicate, even though those versions may not be fully compatible. Numerous methodologies and infrastructures that address these challenges and make it possible to upgrade distributed systems automatically while limiting service disruption have been proposed in the past.

Senivongse [1], [2] describes how to use mappers to enable cross-version interoperation during distributed upgrades. A mapping operator is a mediator object that upgrades an old-version request to a request for the new service. To be able to intercept the request without alteration on the underlying invocation protocol, the mapping operator may disguise as an instance of the old service. By not knowing that the old service has evolved, the client transparently binds to the mapping operator and sends a request to it. The mapping operator does not execute the request but upgrades the request and diverts it to an instance of the new-version service.

Liskov et al. presents an ontological approach to describe dependencies in the software system [3]. She defines different relationships and constraints of software components. The authors claim that their method supports reasoning that is needed to ensure that programs that work correctly using the supertype component continue to work correctly with the subtype.

B. Compatibility checking of evolving services

A major advantage of Service-Oriented Architectures (SOA) is composition and coordination of loosely coupled services. Because the development lifecycles of services and clients are de-coupled, multiple service versions have to be maintained to continue supporting older clients. Typically versions are managed within the SOA by updating service descriptions using conventions on version numbers and namespaces. Several automated methods to evaluate compatibility among services have been created.

Becker et al. [4] propose to verify the compatibility among services on the basis of a Versioning framework modelled in UML. A service is represented within the SOA by a model schema, which defines the external representation of the service as a set of versioned abstractions and relationships between those abstractions. Authors refer to these abstractions collectively as types. The service model defines the collection of types exposed by the service within the SOA. When a service implementation evolves, changes are reflected in the corresponding service model with the update, addition or removal of types that represent updated, added or removed service functionality.

C. Feature interaction in telecom systems

In the early 1960s, switching systems began to rely on software control for the establishment, maintenance, and removal of connections necessary to complete telephone calls. Since then, features (i.e. modifications of or enhancements to the control of telecommunication services) are added by modifying or enhancing this control software. Mature switching systems can contain hundreds of features. Unfortunately, as the number of features grows, so does the time needed to introduce new ones. A key reason is that new features may interact in unexpected or adverse ways with existing features. These adverse or unexpected interactions form the essence of what has become known as the feature-interaction problem. This problem has been studied in 1990s and numerous methods for detection of feature interaction have been proposed.

Stepien and Logrippo [5] detect a category of interactions which manifest itself in ambiguous actions, i.e., nondeterministic behavior of the 'black box' being specified like for example the situation where the user presses a button, and one of two different effects can follow. LOTOS [6] is used as a formal description technique.

Inoue et al. [7], [8] detect interactions which manifest themselves by nondeterministic state transitions, by the absence of an executable state transition (deadlock), and by transitions to so called "abnormal states" i.e. logical contradictions in the relationship between the state of terminals, using the State Transition Rules (STR) description technique. Interactions are resolved by adding rules which select one of the nondeterministic transitions or introduce a new state transition, respectively.

Methods developed in the above studies cannot be directly applied to solve the problem of feature compatibility in telecom domain where an automated system with a possibility of human control would be a more suitable solution. In the next section we describe in more detail the problem with current approaches to the feature compatibility problem.

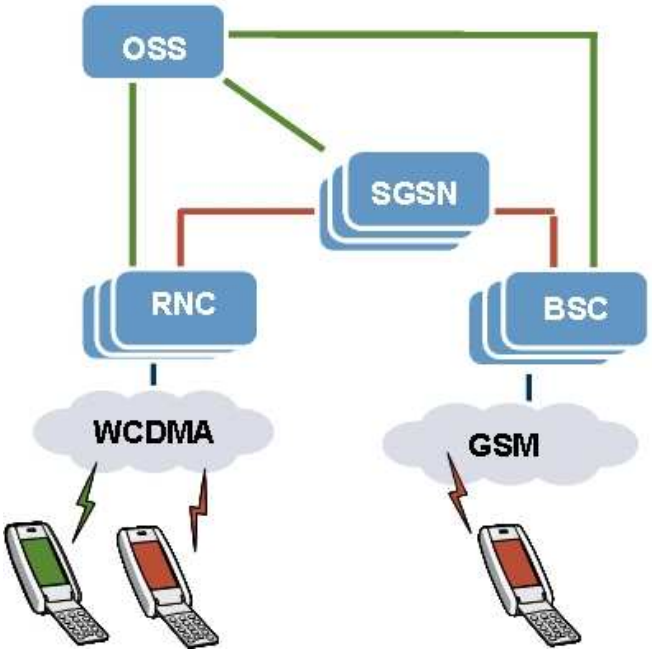

Figure 1. Nodes involved in SGSN In Pool feature.

III. Problem formulation

An upgrade or deployment of the network is preceded by a study of the compatibility between network components and their versions. In addition, network features have specific constraints and requirements that must be fulfilled before the activation of a feature e.g. the SGSN In Pool feature requires that SGSNs in a Pool are at the same version level. The process of checking compatibility prior to activation of a feature is manual and is performed by a network expert. Therefore it is a time-demanding, error-prone and expensive activity. To make matters worse this process must be repeated every time a new feature is deployed or upgraded.

Currently, an expert uses a compatibility matrix to check the compatibility between different software releases and network interfaces. It is a two-dimensional matrix[6] that specifies the node combinations with old and new releases and the interfaces, which must be compatible. A compatibility matrix tracks two levels of compatibility. The first level of compatibility could be termed "implemented compatibility". This is where compatibility has been designed into specific versions of nodes at either side of an interface. The second level of compatibility can be termed "tested compatibility". This is where compatibility of an interface has been tested and verified for an interface between specific versions of two nodes.

The matrix is usually updated when a new version of a node is released. Then the expert can use it every time he plans an upgrade or deployment of a network feature. Because the network feature spans across multiple nodes, the compatibility matrix can be used to check the consistency between software versions of nodes which have a direct dependency. Consider the example of SGSN In Pool feature shown in Figure 1.

SGSN In Pool makes it possible to connect a BSC or RNC to several SGSNs. It introduces a more flexible and efficient architecture with built-in network redundancy, which replaces the traditional hierarchical network structure. A specific SGSN In Pool for WCDMA feature involves interaction between several RNCs, SGSNs and OSS nodes. Before an activation of SGSN In Pool feature, an operator must make sure that the versions of nodes interacting in the feature are compatible.

In addition to checking the inter-node compatibility, an expert must verify that feature-specific constraints are fulfilled in order to activate a feature. This information is usually provided in the documentation describing the feature. An example of such constraints is the requirement that, for the "SGSN in Pool"feature, all SGSNs in a pool must be of the same version level.

The current solution used to check feature compatibility is not only cumbersome but also it is not future-proof. One of the main objectives behind the design of the 3GPP System Architecture Evolution (SAE) was to develop a new core network architecture for Universal Mobile Telecommunications System (UMTS) that addresses cost-efficient deployment and operations for mass-market usage of IP services as well as improvements in integration of other access technologies in the network [9]. However the design and deployment phases of SAE are expected to be more complex to deploy than 3G core network.

In addition, in the near future the network will be composed of equipment from two or three different technologies (2G, 3G, LTE). The software update of such networks will be more complex and will require more expertise than a network composed of a single technology.

In the future, the core network may be composed of equipment provided by different vendors. Because mobile networks exist for many decades, there will be less new network roll outs and the network vendors will have to integrate new features and equipment with resources already deployed. Upgrade of features and software in such heterogeneous environments will require a lot of expertise, effort and time. In our solution we address the problem of feature compatibility in current and next generation mobile

[6]The matrix is often built in a spreadsheet program such as Excel.

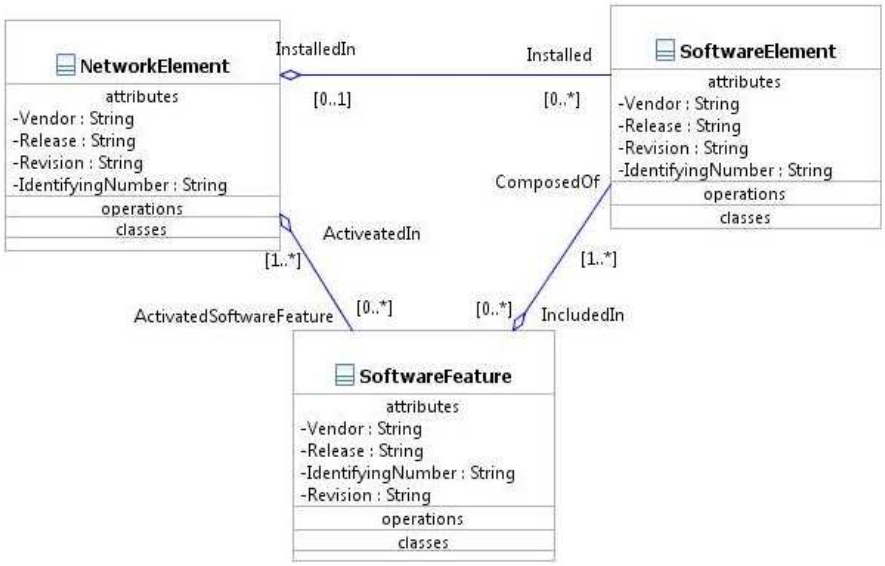

Figure 2. The meta-model of features, software and network elements defined using UML

networks.

IV. Solution overview

In order to automate feature upgrade and deployment in telecom networks we propose to introduce a new method. This method may be supported by a tool, a prototype of which is described in section VI. The proposed method is a two-phase process.

In the first phase the system reads a model prepared by an expert of the network feature. The model includes the requirements and dependencies of the feature on other features and software units (see Figure 3). The model is

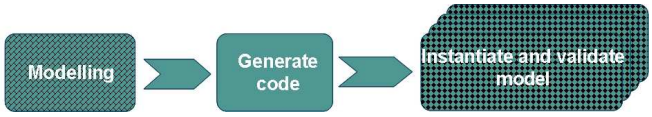

Figure 3. Three phases of the proposed method for feature compatibility assessment.

composed of two parts detailed in section V. The first component is a UML meta-model that represents general network components (like network elements or software elements), their properties (e.g. release) and relationships (e.g. InstalledOn). The second component is expressed by OCL constraints that apply to a specific feature. This can describe compatibility between software and network release or required type of network elements in a feature.

In the second phase the model can be reused over and over again for assessment of the compatibility of the feature for different network instances. When a representation of the network resources is instantiated, it includes information about deployed (or projected) network entities, connections between them and software units. This information is combined with the models of network features previously built. As a result, a compatibility matrix is automatically composed which contains information about the existing (or projected) features and their dependencies in the considered network instance. Our method traverses the compatibility matrix in order to verify if all the requirements provided in the features' models have been fulfilled in the considered network instantiation.

If the assessment is successful, the user can deploy the projected features according to the instantiated representation of the network verified with the method. Otherwise the method proposes modifications to the network resources in order to fulfill feature requirements.

Between the modelling and assessment phases, an intermediate step of code generation is required (as shown in Figure 3). In this step the information from the model is transformed into an application that can be executed by a user who wants to assess feature compatibility in a network instantiation.

V. Detailed description of the solution

To realise and demonstrate the new automated process described in section IV we developed three components: a model, a method and a prototype that allows automatic verification of end-to-end compatibility among network features.

A. Feature model

The feature model enables common representation of features and their dependencies in a telecom network. This information can be then used to verify end-to-end compatibility among network features. The proposed model

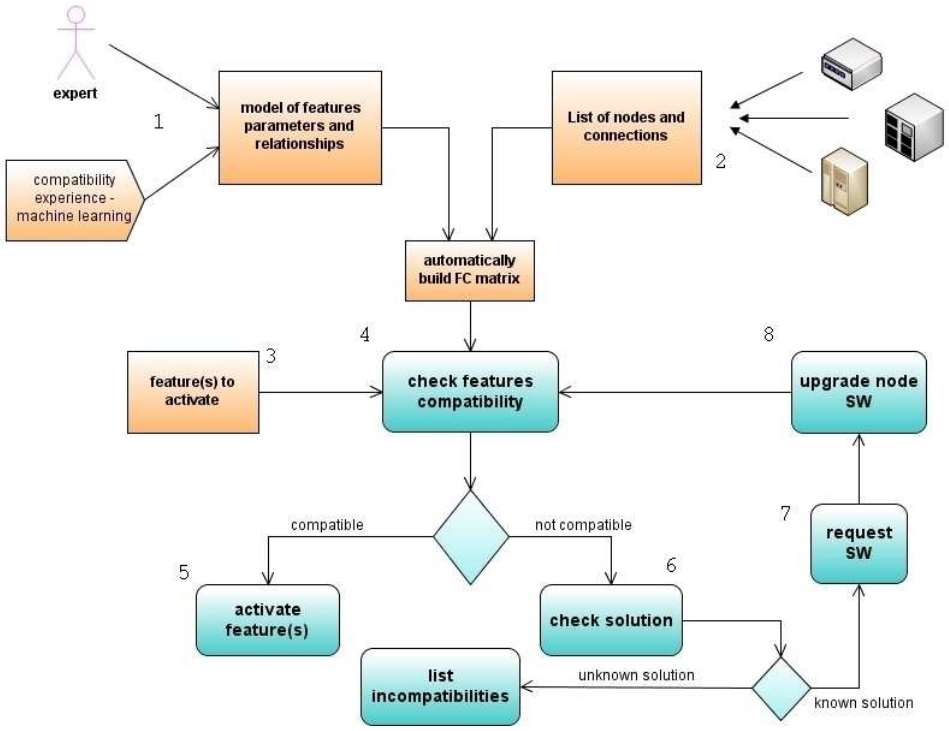

Figure 4. Proposed method of automatic feature compatibility assessment.

is composed of two parts: UML meta-model and OCL constraints [10].

1) Meta-model: Figure 2 shows a meta-model designed using UML that represents elements that have an impact on the feature compatibility in the telecom network. The NetworkElement class corresponds to a logical entity in telecom network (e.g. SGSN or RNC).

The SoftwareElement class represents a software package that provides specific functionality on a network element. This could be for example IP connectivity or Support for SGSN In Pool on RNC. SoftwareElements can be installed on NetworkElements which is expressed by the relationship Installed/InstalledIn.

The SoftwareFeature class models network functionality, usually deployed over several nodes. This could be for example SGSNInPoolForWCDMA which involves interaction between several SGSNs, RNCs and an OSS. This interaction is modelled by the ActivatedSoftwareFeature/ActivatedIn relationship. Also, SoftwareFeature may be implemented in several SoftwareElements which is represented by the relationship ComposedOf/IncludedIn.

These classes have similar properties (Vendor, Release, Revision and IdentifyingNumber) used for the assessment of feature compatibilities. The Vendor property provides information about the provider of the corresponding class. This information will be explored in particular for the multiple-domain networks. The Release and Revision properties are used to distinguish different versions of modelled entity. These parameters are important for the assessment of compatibility because the compatibility between different types of nodes is version-based. Finally the purpose of the IdentifyingNumber property is to uniquely identify the modelled entity.

This meta-model can be extended to represent specific network features, network and software elements. The above meta-model is decorated with OCL constraints which describe feature-specific requirements.

2) OCL constraints: The Object Constraint Language (OCL) [11] is a commonly used declarative language for describing constraints (rules) that apply to UML models. These constraints are the conditions that must be true about some aspect of the system and can be automatically verified on an instantiation of the UML model. The network feature can have different constraints that are grouped in three categories:

- Generic requirements for the network feature. These constraints are specific to the network feature. It may be a requirement about existence of other features, software elements or network elements. In case of SGSN In Pool it is the requirement about SGSN nodes included in the feature.
- Requirements about software elements and their version needed on the network elements. In order to support a feature the network element requires specific software

Figure 5. Intermediate steps to assess feature compatibility.

elements. In case of SGSN In Pool an RNC node requires specific version of software element that supports SGSN In Pool functionality.

- Constraints about the compatible version of nodes that are present in the software feature. For the SGSN In Pool scenario it may be a condition that SGSN Release "X" Revision "x" is compatible with RNC Release "Y" revision "y".

Once the OCL constraints are created they can be automatically validated against various instantiations of the UML model. The validation process consists of traversing the feature model and applying the OCL constraints to the appropriate entities.

B. Compatibility assessment mechanism

The new method applied during the "Instantiate and Validate model" step from Figure 3 is detailed in Figure 4.

The method relies on a model of features and their relationships provided by an expert. The model could also be built on the basis of the previous experience using machine learning (1 in Figure 4). In addition to that network elements of a specific network instance send information about their current software packages (2). The system automatically builds a feature compatibility matrix which represents the information about the compatibility between the existing components of the assessed network. A user can select a new feature or a new version of an existing feature to be activated (3). The tool will automatically check if the compatibility requirements are fulfilled (4). If the verification is successful, the features will be activated (5). Otherwise the system will look for a solution (6) and will request the vendor for the required software (7) or provide the list of incompatibilities if it cannot find the solution. The system will be informed about each software upgrade by the relevant node (8) and if there are any features waiting for activation, it will check compatibilities and eventually activate the features.

The proposed process may be partially or fully automated: it can be fully controlled by a user who can approve the activation of a network feature. The interaction of a network expert is minimized in comparison to the current process. The expert's expertise is needed only once, at the beginning of the process to model a network feature and its prerequisites. The expert's model is reused over and over again by the automated tool whenever an instance of the modeled feature needs to be activated. In the current approach, the expert is required every time a new feature is activated.

VI. Prototype of the solution

The prototype of a system that demonstrates the new automated method of feature compatibility assessment is based on the Eclipse platform and is composed of several modules (plug-ins). A schema of the application is shown in Figure 6.

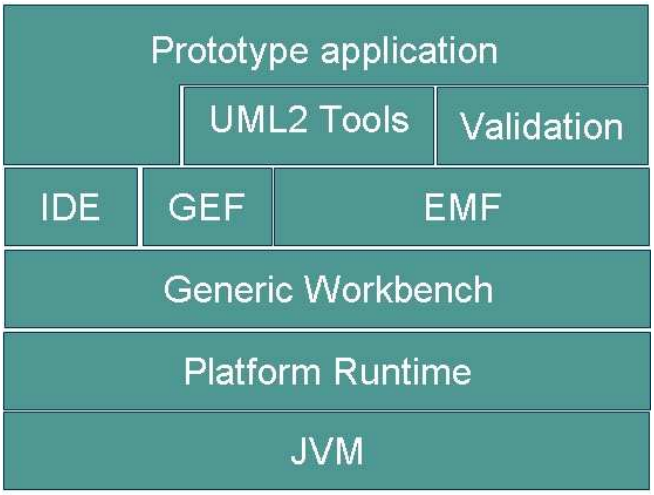

Figure 6. Schema of the prototype application.

The application is Java based and therefore runs on top of a JVM and an Eclipse Platform Runtime which is responsible for the activation and management of the application modules when they are required. The prototype runs as a Generic Workbench application. Workbench is the term used for the generic Eclipse UI [12].

The application requires an Integrated Development Environment (IDE) to display and eventually modify the code generated by the prototype. Automatically generated code can be modified and to simply this task the Eclipse IDE provides several functions like source code editor, build automation tools and debugger.

The Graphical Editing Framework (GEF) module enables a graphical representation of the feature components. GEF is used by the expert to model a feature in a tree-based editor or UML class diagram. Also an expert uses GEF functions when he represents network instances in tree-based editor.

Eclipse Modelling Framework (EMF) is the core component of the prototype. It provides the functionality of Unified Modelling Language (UML) modeling and Object Constraint Language (OCL) constraints generation. It also enables the generation of the Java code on the basis of the

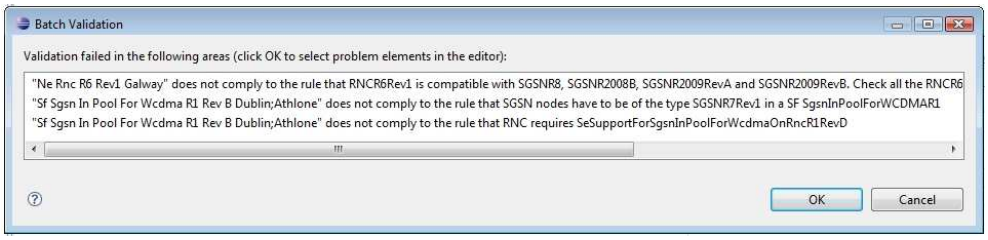

Figure 7. Proposed solution to the SGSN In Pool feature incompatibilities.

UML model. Finally, EMF allows the instantiation of the UML model.

A UML model can be transformed into a class-diagram with the UML2 Tools plug-in. An expert can work with a tree-based editor and a class-diagram to build a feature model. UML2 Tools assures synchronisation between these two visual reresentations.

The Validation framework [13] is used to apply the rules expressed in OCL to an instance of the UML model. When a user decides to check compatibility of a network feature, the validation framework traverses the representation of a network instance and applies the predefined rules to appropriate feature components. Whenever a rule is broken, the framework raises an exception and starts an action predefined by the expert.

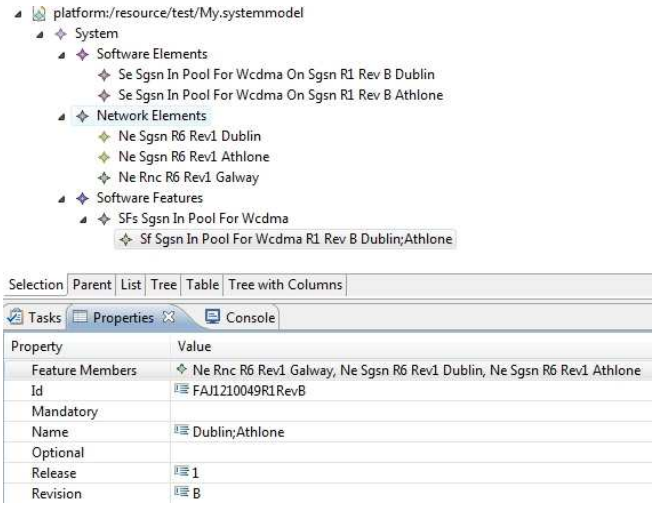

Figure 8. SGSN In Pool feature represented in the prototyped tool.

The prototype can be used by an expert for modeling a network feature (during the first phase of the process described in section V) and by a user to assess compatibility among network features in an instance of a network (during the last phase of the proposed process). Figure 5 shows the intermediate steps performed by the prototype in order to assess compatibility of a network feature.

First, an expert models the network features in UML, their constraints and rules in OCL. The expert then automatically generates a corresponding Ecore model [13] (which is an internal EMF representation of the UML model) which in turn, is used to generate Java code (Java classes) automatically. This code includes a representation of the UML model in Java, an application to graphically generate an instance of the model and a validation engine that is responsible for the application of the predefined OCL rules to the instantiation of the model.

The generated code is an automatically generated application that is executed by a user in order to hold a model of the software features in a particular network instance and to check their compatibility before an upgrade or a deployment of a new feature. To do that, the user first creates an instantiation of the UML model that corresponds to the representation of the network for which the compatibility is to be assessed. This instantiation will be composed of Java objects. Once the instantiation is complete, a user can validate software features (Rules Validation) using the rules defined in OCL and check if a new or upgraded feature can be activated.

The prototype system provides information about the incompatibilities of the assessed network feature. If it discovers any inconsistency in the network feature, the user will be provided with the list of solutions that would result in a correct configuration of the network feature.

Figure 8 shows an example of SGSN In Pool network feature described in section III. The SGSN In Pool feature is provided by three nodes (Network Elements): two SGSNs and one RNC. The model also represents two software packages (Software Elements) installed on both SGSNs. The prototyped tool discovered three incompatibilities in this feature in a particular simulated network instance. It also provides a list of modifications (shown in Figure 7) that need to be implemented in order to achieve feature compatibility.

First, the tool discovered that the versions of SGSNs and RNCs in this feature instance are incompatible. Secondly, the modeled instance of the SGSN In Pool feature requires a different version of SGSN nodes. Finally, a software package supporting SGSN In Pool feature needs to be installed on the RNC node. Once these modifications are introduced, the network feature compatibility would be fulfilled.

VII. Conclusion

We proposed a new method of automatic feature compatibility assessment in mobile networks. This method enables easier and faster deployment of a new network or an upgrade of an already existing telecom network. Moreover it can become a part of an automatic license management solution. After a network operator activates new network features the system will automatically recognize which software packages are required on the nodes and request them from the vendor. We also described a prototype of a tool that can be used to support the described method. It demonstrates the possibility of integration of commonly used techniques (UML and OCL modeling) with standard technologies (Java, Eclipse) into the proposed method. This would facilitate the integration and use of the described approach into already existing systems.

References

[1] T. Senivongse, "Enabling flexible cross-version interoperability for distributed services," in *DOA '99: Proceedings of the International Symposium on Distributed Objects and Applications*. Washington, DC, USA: IEEE Computer Society, 1999, p. 201.

[2] T. Senivongse and I. Utting, "A model for evolution of services in distributed systems," in *Distributed Platforms*, S. Schill, Mittasch and Popien, Eds. Chapman and Hall, January 1996, pp. 373–385. [Online]. Available: http://www.cs.kent.ac.uk/pubs/1996/341

[3] B. H. Liskov and J. M. Wing, "A behavioral notion of subtyping," *ACM Transactions on Programming Languages and Systems*, vol. 16, pp. 1811–1841, 1994.

[4] K. Becker, A. Lopes, D. S. Milojicic, J. Pruyne, and S. Singhal, "Automatically determining compatibility of evolving services," *Web Services, IEEE International Conference on*, vol. 0, pp. 161–168, 2008.

[5] B. Stépien and L. M. S. Logrippo, "Feature interaction detection using backward reasoning with LOTOS," in *Proc. Protocol Specification, Testing and Verification XIV*, S. Vuong, Ed. Amsterdam, Netherlands: North-Holland, Oct. 1995, pp. 71–86.

[6] T. Bolognesi and E. Brinksma, "Introduction to the iso specification language lotos," *Computer Networks*, vol. 14, pp. 25–59, 1987.

[7] Y. Inoue, K. Takami, and T. Ohta, "Method for supporting detection and elimination of feature interaction in a telecommunication system," in *Proc. International Workshop on Feature Interactions in Telecommunications Software Systems*, Dec. 1992, pp. 61–81.

[8] Y. Inoue, T. Ohta, and K. Takami, "Automatic detection of service interactions in telecommunications service specifications," in *Communications, 1994. ICC '94, SUPERCOMM/ICC '94, Conference Record, 'Serving Humanity Through Communications.' IEEE International Conference on*, May 1994, pp. 1835–1841 vol.3.

[9] Ericsson, "LTE an introduction," White paper, June 2009, available online (16 pages). [Online]. Available: www.ericsson.com/technology/whitepapers/lte_overview.pdf

[10] OMG, "Technical guide to Model Driven Architecture: The MDA guide," OMG paper, June 2003, available online (62 pages). [Online]. Available: http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

[11] "Object modeling with the ocl, the rationale behind the object constraint language," London, UK, 2002.

[12] J. McAffer and J.-M. Lemieux, *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java(TM) Applications*. Addison-Wesley Professional, 2005.

[13] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley Professional, 2009.