A Framework for Model-Driven Proposal and Evaluation of TABAC Strategies

Jochen KÖGEL[1], David LUTZ[2], Marc BARISCH[1], Yongzheng LIANG[2]
[1] *University of Stuttgart, Institute of Communication Networks and Computer Engineering*
Email: {jochen.koegel, marc.barisch}@ikr.uni-stuttgart.de
[2] *University of Stuttgart, Computing Center*
Email: {lutz, liang}@rus.uni-stuttgart.de

Abstract: Trust, Accounting, Billing, Auditing and Charging (TABAC) are crucial functions of productive IT and telecommunication systems. However, they are often not considered in early stages of development, but have considerable impact on commercial success at later stages. On the one hand, these functions are costly to be added after having a running prototype. On the other hand, they should not be burdens in early development stages. In order to solve this dilemma, we propose a framework for model-driven proposal and evaluation of TABAC strategies. This framework allows both: evaluating the possibilities to add TABAC functions at an early development stage and adding TABAC strategies to running prototypes by using existing interfaces and applying general models from a model pool. The framework not only proposes suitable strategies based on capabilities and requirements of the existing system. Moreover, it aims at instantiating the TABAC functions in a runtime environment for feeding them with real data from the existing system. This outstanding feature allows compiling a proposal for the best suiting strategy based on realistic interactions with the prototype.

Keywords: Model-driven design, Accounting, Billing, Auditing, Charging

1. Introduction

Today, business in the IT-Service and Telecommunication sector is subject to rapid changes and requires a flexible computing and communication infrastructure. It is key to adapt the existing infrastructure to new requirements and to enable the fast introduction of new services. In particular time to market is in many cases the crucial factor that determines on commercial success of newly introduced or adapted services. Therefore, it is required to have efficient development and management processes.

The typical development process, however, is time consuming and involves the creation of several prototypes and testing phases before moving new services to production systems. In many cases prototypes only show the basic feasibility and integration into the existing infrastructure, but often neglect features that are important for productive operation. Among these often neglected features are trust, accounting, billing, auditing, and charging (TABAC). Adding such features in late development stages delays the introduction of new services even more. On the other hand developing such functionality in prototypes at an early stage is questionable, since the prototypes might not make it into production.

Therefore, we propose a concept that allows the introduction of TABAC functionality in a very late stage of the service development process or even after the service has been deployed. That means developers can focus on the actual prototype development to improve time to market. Hereby, we have two application scenarios in mind (Sec. 1.1), leading to a couple of requirements (Sec. 1.2) and key aspects of our concept (Sec. 1.3).

1.1 Application Scenarios

Many different application scenarios exist where our concept might become advantageous. Throughout this paper, we will restrict ourselves to two application scenarios, which we consider as important: Future Internet testbeds and existing IT-service infrastructures.

Future Internet testbeds: In recent years a couple of Future Internet testbeds [1,2] have been built or are in development [3]. These testbeds are in most cases focused on the provision of testing platforms and the empirical evaluation of new protocols and services. Therefore, TABAC functionality is often neglected in the testbed itself and in the protocols.

Existing IT-service infrastructure: Due to changed regulations or new requirements with respect to charging, it might become necessary to introduce TABAC functionality into existing systems. An example for such a transition is given in Sec. 5.

1.2 Requirements

Our concept has to be flexible, extensible and cost-efficient in order to improve the service development process. Moreover, it must be even usable by other persons than the service developers and provide adequate feedback about the configured system.

A high degree of flexibility means first of all that we have to deal with a plethora of different TABAC strategies. For example in case of charging we have to differentiate between offline and online charging. In particular the latter one might have an impact on the prototype. That means we need some kind of repository that provides the different strategies. Moreover, our concept must be adaptable to different prototypes and service infrastructure, i.e. different architectures and protocols are in place that need to be abstracted in an appropriate way. It must also be possible to compare the impact of different TABAC strategies on the prototype itself and on the generated revenue. That means our concept has to comprise evaluation possibilities.

Since prototypes as well as TABAC strategies are subject to change over time, it must be possible to *extend and adapt* the components in our concept in an easy way. For example, if a new charging model comes up or the regulation changes it must possible to provide new components to support the changes.

In addition to faster development times, we have to ensure that our concept is cost efficient. That means that only *minimal changes within existing prototypes* and service infrastructures are necessary for the introduction of TABAC functionality. Moreover, our concept must not tremendously impact the performance of the prototype itself.

Moreover, our concept has to provide a high degree of *usability*. It should be possible that non-experts make use of our concept. That means that a graphical user interface is required that allows the combination of different TABAC strategies with the prototype under consideration.

Finally, we have to *evaluate* the combination of TABAC strategy and prototype with respect to the feasibility, the potential revenue, and the costs created by introducing the TABAC strategy.

1.3 Overall Approach

Our approach intends not to be invasive to the existing infrastructure, but being an add-on for proposing and evaluating TABAC strategies (see Fig. 1). Based on the development process of the prototype, the administrators and developers select the interfaces that can be used for retrieving the required data. The description of these interfaces together with their capabilities and dependencies is passed along with the requirements for the TABAC solution to our framework.

The framework itself (see Fig. 1, right) contains information and dependency models that include existing standards and is extensible by adding prototype-specific information. Based on the existing and prototype-specific models, the framework proposes several suitable strategies. These strategies are then put into a runtime environment (RE), where they process live data gathered from the prototype interfaces. This step allows the evaluation of different strategies based on real data with typical characteristics from the prototype in order to find the most suitable approach. The evaluation result can not only be used to find the best suitable strategy, but also serves as input for prototype developers and administrators (dashed arrow in Figure 1).

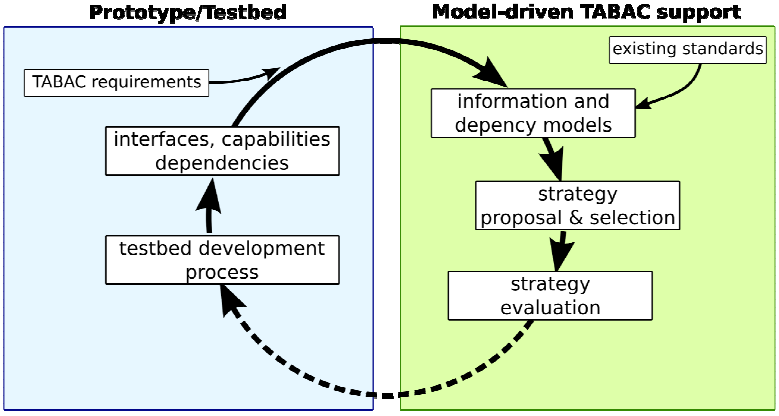

Figure 1: High level view of the framework and the workflow

In order to realize this approach with the requirements introduced before, we have selected an approach based on a couple of key features.

- Model-based: Since we have different TABAC strategies as well as very different prototypes and service infrastructures, we have based our concept on models. Abstract models represent the different TABAC strategies that can be instantiated and combined with prototypes, which are as well described by models.
- User Interface: Our concept is based on a graphical user interface that allows the user to evaluate different TABAC strategy combinations and get feedback on the potential results by real-time evaluation.
- Tool support: The creation of models and the corresponding instantiation is a complex task. Since we do not want to build everything from scratch, our concept is based on existing modelling tools like EMF [4] or IBM Rational Tau[5].
- Runtime environment (RE): For the interworking between the models and the prototypes, we make use of an RE. The RE allows the placement of instantiated models and let them work on real monitoring data. Based on the resulting data, we can provide the above introduced evaluation possibility.

1.4 Structure of Paper

The remainder of this paper is structured as follows. Sec. 2 reviews the related work in the area of modelling and in particular what can be achieved with modelling at runtime. Sec. 3 introduces the overall framework to achieve the introduced requirements and to provide the key features. We illustrate the relevant workflows in Sec. 4 and provide an exemplary study of the application of the proposed framework and workflows for a VoIP prototype in Sec. 5.

2. Related Work

Since we do not want to enhance TABAC functionality on its own, but rather employ existing standards and protocols, we focus on related work in the area of models and in particular on models at runtime. Hereby, a model in terms of Model-Driven Engineering (MDE) is

an abstraction of some aspect of an existing or planned system. Models are used to describe complex systems at multiple levels of abstraction and from a variety of perspectives. MDE is typically used to describe software development approaches in which abstract models of software systems are created and transformed into concrete software implementations.
In the last years we have experienced the emergence of new applications classes, which are highly complex, inevitably distributed, and operate in heterogeneous and rapidly changing environments. These systems are required to be adaptable, flexible, and reconfigurable. MDE has focused primarily on using models at design, implementation, and deployment stages of development. It is limited to meet the new challenges. The concept of Models at Runtime [6] extends the applicability of MDE techniques to runtime and makes use of modelling techniques beyond the design and implementation phase. System users can use runtime models to observe and monitor the runtime behaviour. A runtime model can also be used to dynamically adapt the model within running systems [7]. A key benefit of runtime models is that they provide a richer semantic base for runtime decision-making related to system adaptation, configuration, validation and other runtime concerns.
The EU FP7 Project DiVA [8] provides an integrated framework for managing dynamic variability and unifies runtime adaptation and runtime evolution by monitoring both the run-time platform and the design models. The paper [9] presents the modelling of accounting components both at business level and at network level. The accounting components are responsible for correlating network and service level accountable events, resulting in a single bill for the service subscribers. The white paper [10] depicts an information model for a federated accounting management system.

3. Framework

Our framework provides model-driven development and proposal of TABAC strategies. However, this is not limited to creating models based on a model pool, but also includes a RE for connecting the models to the interfaces of the prototype and evaluation of TABAC strategies with live data.

3.1 Overview

Our framework supports a workflow for selecting suitable TABAC strategies and evaluating them using live data from prototypes. This workflow is indicated in the overall framework overview in Fig. 2 by the dotted arrows. It starts with collecting interface properties and TABAC requirements serving as input for the strategy proposal and selection that is based on an extensible model repository. The Instantiation of the models leads to a representation of the TABAC strategy in a RE, which allows the collection of measurement data for strategy evaluation. The result serves as input to the prototype provider for identifying suitable strategies and possible improvements. Sec. 4 further details this workflow.

In order to support this workflow, our framework relies heavily on models that reside in a model repository, which is the core of the first building block: the *Model Layer* (Fig. 2). This Model Layer also contains the Strategy Proposal & Selection component, the model Instantiation and the Strategy Evaluation. The second building block is the *Runtime Layer* that evaluates instantiated models using live data from the prototype and delivers measurement data to the Strategy Evaluation. In addition to these two layers, the interfaces to the prototype form a third part. These are not only protocol and monitoring interfaces, but also interfaces for TABAC descriptions and results as well as user interfaces for the evaluation process.

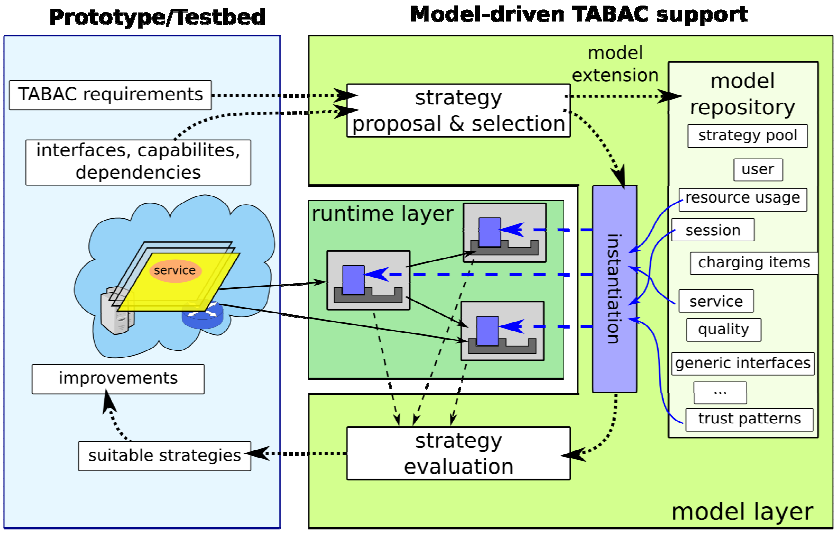

Figure 2: Detailed Framework overview

3.2 Model Layer

The purpose of the Model Layer is providing generic models for TABAC concepts and strategies with several concrete refinements, such as support for certain protocols or session concepts as a basis for instantiation of strategies and their evaluation. Besides these concepts, the model repository also contains a pool of TABAC strategies and patterns for different trust models. For creating TABAC strategies, the present models are extended by definitions from the prototype. The goal is to limit the amount of models and knowledge that has to be added manually to a minimum.

Based on the strategy selected, the required components are taken from the model repository. The Instantiation block puts them together so that the required processing steps are defined and feeds the model with prototype specific parameters. The resulting description of a TABAC strategy is then put into the Runtime Layer for evaluation.

3.3 Runtime Layer

In order to evaluate TABAC strategies with live data from the prototype, the Runtime Environment (RE) takes instantiated models, connects them to the prototype and performs measurements for the strategy evaluation. This includes transformation of protocol-specific data models to generic concepts that are aligned with the items of the model repository.

While in a straight-forward approach the RE collects monitoring and accounting data, we envision also active communication of the RE and models with the prototype. This might be necessary for certain strategies, where additional user-specific records from AAA servers are required. Also prepaid charging models demand for interaction with the prototype for tearing down sessions when users run out of credits.

3.4 Interfaces towards the prototype

A critical part of our framework are the interfaces towards the prototype and its administrators. As shown in Fig. 3 on the left, this includes interfaces for collecting data that will be processed. Such data can be delivered using well-known protocols, such as SNMP, IPFIX or DIAMETER, but also yet unknown prototype-specific protocols. Thus, our approach is to map data delivered by these protocols to general concepts of our models in the prototype interface part in order to be independent of protocol peculiarities.

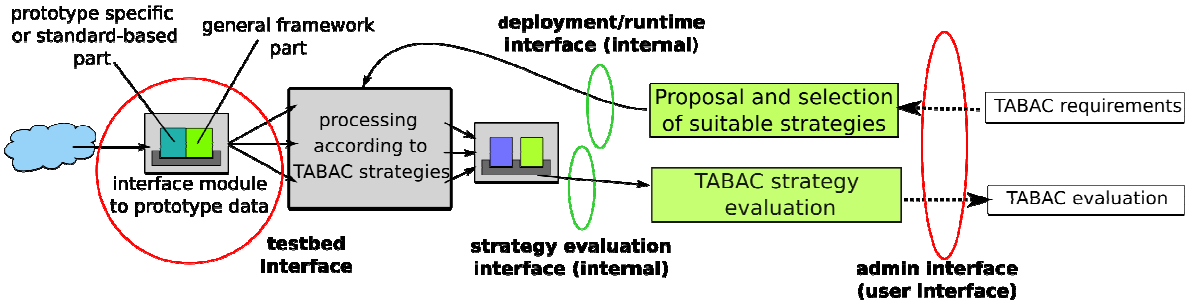

Figure 3: Interfaces of the framework

Internal interfaces that have to be obeyed by the RE concern the deployment and strategy evaluation interface. Here, it is vital to transform the deployment description and measurement data to a common format to get consistent evaluation results.

For reaching our goal of usability of the overall framework, an admin interface is provided that serves as central dashboard for strategy selection and evaluation. It presents each step of the workflow and drastically enhances the process, since the effort for e.g. a strategy where additional protocol modules are required can already be estimated at an early stage.

4. Workflow

While Sec. 3 has presented the design of the intended framework, this section provides an overview how the framework will be used to connect to prototypes and to enable TABAC models at runtime. The overall workflow can be divided into six steps that have to be carried out. These steps are: Problem Detection, Planning and Identifying the Interfaces, Deployment of TABAC-models, enabling the prototype's Interfaces, Simulation and Measurements, and Evaluation of the data received and the TABAC-models used.

Problem Detection: The first step is the formulation of the problem, which shall be solved by using the TABAC framework. We have identified three different challenges for using this framework. Firstly, the framework can be used to enable TABAC functionalities that have been forgotten when establishing the prototype. Secondly, the framework can be used instead of deploying TABAC elements, e.g., because the prototype's purpose is not related to TABAC elements, so, the use of this framework avoids efforts regarding TABAC components. Finally, this framework can be used to evaluate TABAC strategies. A prototype operator may have deployed TABAC concepts and uses this framework to evaluate them. This framework can then be used to evaluate the TABAC functionalities and to analyze suitable solutions for this prototype. Therefore, before starting to use the framework, the prototype administrator has to formulate the problem, the framework shall address.

Planning: Besides identifying the problem, the capabilities of the prototype and the intended way of proceeding have to be taken into consideration. Thus, after detecting the problem, in a planning phase the procedure of connecting the prototype to the model repository provided by the framework has to be defined. This includes initially identifying the possible interactions of the prototype with the framework's repository, such as available protocols and layers as well as defining the intended TABAC elements.

Model Deployment: After the planning phase has detected the layers, protocols and interfaces intended to communicate with the TABAC models, the model has to be deployed. This is carried out by choosing the required elements from the TABAC model repository provided by the framework. Here, several elements can be combined to TABAC models to be connected to the prototype. However, if the planning phase has identified a TABAC functionality not provided by the repository, the framework offers the opportunity to create new models for this particular case. For deployment of the TABAC models, a user-friendly frontend for the repository will be created.

Interface Enabling: When having established the TABAC model that is expected to be connected to the prototype, the interfaces of the prototype have to be enabled in order to communicate with the model. Therefore, the elements of the prototype that supply the required data identified during the planning phase have to be activated and to be connected to the runtime model to deliver the data.

Measurements: As soon as the model deployment phase has provided a TABAC model and the interface enabling step has connected the model to the prototype, the admin may start the model-driven measurements. To run the demanded processes, the admin can either get data from test-users or carry out the required operations based on models and simulations. The measurement phase has to execute prototype operations in order to collect this data required for the model evaluation.

Evaluation: After the measurement phase has provided enough data, the collected data can be analysed. These evaluation procedures rely on the problem statement defined at the beginning of the process. The first evaluation, which is the same for each of the problem statements, is related to the quality of gathered data. The framework provides evaluation functionalities that are able to discover whether the received data are of such a quality that they can be used for TABAC elements. The second evaluation step focuses on finding the best strategy, evaluate different TABAC concepts used within the prototype and provide the best suiting elements of the deployed TABAC models.

5. Usage Scenarios

In order to get an idea of the functionality of the framework, we consider a scenario where TABAC functions are added to an HD video conference and collaboration prototype.

5.1 From Prototype to productive Service

In our scenario an IT company has developed a HD videoconference and collaboration system as a prototype. Due to its unique features the prototype is highly popular and heavily used for collaboration tasks with other companies. Even other companies ask whether they could use this service for their internal collaboration and they would pay for it. At this point, the question for suitable TABAC strategies arises, since the prototype is already used, has proven its stability and usefulness but lacks TABAC functions.

The administrators discuss with business department how to charge other users and how to include accounting functions into the prototype. With the help of our framework, they can evaluate different strategies with only allowing access to a few interfaces and without changes to the running prototype.

5.2 Requirements and available interfaces

In a first step, the TABAC requirements and interfaces of the service must be defined as input to our framework. The administrator and the business department identify the following requirements

- Users of the service must be known to the company or known users have to provide statements that they trust users they invite.
- Collaboration sessions should always include all participants, even if one of them runs out of credits. Reduced quality is acceptable but has to be logged.
- The charging should consider the resource usage of the collaboration sessions and compensate in a suitable way. What actually is charged for is not put into requirements.

In addition to these requirements, the administrator compiles as list of available interfaces jointly with the developers. The following interfaces are available:

- IP traffic accounting to the endpoints of the media streams (e.g. IPFIX)
- Charging interface to the company's billing system
- Session details from the video conference service (IP address of participants, duration of sessions, video quality). This data is available once the sessions are terminated.
- Control interface to the system that allows for setup and teardown of sessions.

5.3 Proposed strategies

Based on the input from the administrator, our framework identifies the following three strategies as first candidates for an evaluation:

- *Strategy A "Postpaid with detailed CDR":* Users are charged according to duration and amount of traffic caused using the collaboration service. The framework detects that the information for correlating CDRs with traffic information is only available after session termination, thus this must be a postpaid strategy.
- *Strategy B "Prepaid based on quality level and duration":* The framework suggests using a prepaid module from its model repository where credits can be bought using the companies billing system. When a user runs out of credits, the session is typically terminated with this model.
- Strategy C: "Pay per use": Each time the collaboration service is used, a fix amount of money is charged from the user using the company's billing system.

5.4 Result

Based on the three proposals, the administrator selects the three strategies for instantiation and test in the RE. Over a certain time the strategies are evaluated while the prototype is used as usual. The strategy evaluation block then compiles results for the three strategies

- *Result Strategy A*: The postpaid strategy is able to interact with the company's billing system to correctly bill the users. However, the CDRs are sometimes incomplete and correlating traffic accounting data with end users is often imprecise.
- *Result Strategy B*: Using a prepaid strategy works in this case, however, tearing down the session in case of empty credits of one user violates the requirements. The framework suggests extending the control interface to enable quality degradation in this case.
- *Result Strategy C*: Pay per use in principle works, but the amount charged from users is not aligned with the resource usage caused, leading to implausible bills. This results from usage and traffic properties, which could only be determined using real data.

The framework decides that Strategy B is the most suitable one, even if minor changes at the service interfaces are necessary. In our scenario, this strategy was not initially in focus, but proved to be realistic by putting the TABAC strategy into work in the RE.

5.5 Advantages of this Procedure

The approach described in this paper can be used to achieve even more benefits, such as ignoring TABAC elements and focusing on research topics. Since this approach will define suitable TABAC elements at the end, the company taken for this example scenario benefits from the large TABAC repository, the TABAC element evaluation and the intactness of the running prototype. Firstly, the company has no need to gain knowledge about all possible TABAC strategies, since the repository contains a huge pool of different solutions. Here, lots of resources are saved. Secondly, predefined algorithms are able to choose the best fitting solutions out of a pool based on measurements taken with the prototype. Thus, the

company does not have the need to compare all the results and to select, possibly faulty, one of them, since the algorithms will present a list of the best ones. Thirdly, the prototype does not need to be touched, except of defining and configuring the interfaces to the TABAC framework. By following the concept of defined, non prototype-specific TABAC models, no add-ons are required for the evaluation. Thus, the prototype can remain unchanged and even in operation.

6. Conclusions

We introduced a framework for model-driven selection and evaluation of TABAC strategies. It allows administrators to evaluate TABAC strategies for already existing prototypes in order to turn them into productive commercial systems. Additionally, the framework can provide feedback for developers. Our example scenario demonstrated that the process need not stop at proposing strategies. Moreover, the real value lies in putting the models to work in a Runtime Environment using real data and interfaces. This leads to a comprehensive evaluation with high value for administrators and developers. We plan to prove the applicability of this approach by implementing this framework and demonstrate its capabilities by interfacing to real prototypes.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr. 215832 (SWIFT).

References

[1] PlanetLab – An Open Platform for Developing, Deploying and Accessing Planetary Scale Services, www.planetlab.org

[2] Szegedi, P et al.: With evolution for revolution: managing FEDERICA for future internet research, IEEE Communications Magazine, Volume 47, Issue 7, 2009

[3] German Lab – National Platform for Future Internet Studies, www.german-lab.de

[4] Eclipse Modeling Framework (EMF), http://www.eclipse.org/modeling/emf/

[5] IBM Ratioanl Tau, http://www.eclipse.org/modeling/emf/

[6] R. France, B. Rumpe: Model-driven Development of Complex Software: A Research Roadmap. Future of Software Engineering (FOSE'07)

[7] Franck Fleurey, Vegard Dehlen, Nelly Bencomo, Brice Morin, Jean-Marc Jézéquel: Modeling and Validating Dynamic Adaptation, Models@run.time Workshop, Toulouse, France, 30 September 2008

[8] EU Project DiVA http://www.ict-diva.eu/DiVA

[9] P. Hellemans, C.Redmod, K.Daenen, D.Lewis: Accounting Management in a TINA-Based Service and Network Environment. Proceedings of the 6th International Conference on Intelligence and Services in Networks, pages: 13-24, ISBN: 3-540-65895-5

[10] B.Bhushan: White Paper: Federated Accounting Management of Service Usage in a Business-to Business Environment, IST-1999-10357/UCL/WP6/0930-V1