Choreographing security and performance analysis for Web services (Long version)

Stephen Gilmore[1], Valentin Haenel[1], Leïla Kloul[2], and Monika Maidl[3]

[1] Laboratory for Foundations of Computer Science, The University of Edinburgh, Scotland.
[2] PRiSM, Université de Versailles, 45, avenue des Etats-Unis, 78000 Versailles, France
[3] Siemens AG, CT IC3, Otto-Hahn-Ring 6, 81739 München, Germany

Abstract. We describe a UML-based method which supports model-driven development of service-oriented architectures including those used in Web services. Analysable content is extracted from the UML models in the form of process calculus descriptions. These are analysed to provide strong guarantees of satisfactory security and performance. The results are reflected back in the form of a modified version of the UML model which highlights points of the design which can give rise to operational difficulties. A design platform supporting the methodology, *Choreographer*, interoperates with state-of-the-art UML modelling tools such as Poseidon. We illustrate the approach on an example.

1 Introduction

Web services must deliver secure services to users in order that financial and other confidential transactions can be conducted without interference. Off-the-shelf solutions are not available. Web services need to build end-to-end security from the point-to-point security afforded by standard network protocols. Even if a secure system can be created, scaling up to large user populations provides a steep challenge. The availability of many different forms of assistance (caching, stateless session beans, process isolation and others) means that the challenge of building scalable systems is complicated further by difficult-to-quantify approaches to system performance tuning.

We have developed a design platform, *Choreographer*, which seeks to assist with the development of secure systems with quantified levels of performance. To provide an accessible entry point for practising Web service developers the methodology which we support uses the UML. This is a novel feature of our work: we use a modelling language where a specification language or process calculus might more often be used to initiate the analysis. Many UML designs are not analysed either qualitatively or quantitatively. Here we provide support for both types of analysis, and illustrate the value of the analysis via an example.

We use a range of UML diagram types to express the security and performance considerations of the system. As a principle, we use standard UML notation: there are no notational extensions or additional diagram types. This decision has two beneficial consequences. First, a UML modeller using this methodology does not need to learn any supplementary notation. Second, we are able to use standard UML tools such as Poseidon [1] to edit the UML diagrams which we use.

We use class diagrams, collaboration diagrams, sequence diagrams and state diagrams to describe the system under study in UML terms. Additional diagram types may be used in the UML project which is accepted as an input to Choreographer. These can be used for other purposes in model-driven development, such as automatic code generation, and will not interfere with the analysis process. Our aim is to disrupt existing model-driven development approaches as little as possible while adding value to the UML modelling work which would be going on in any case.

Different models can be used for different purposes in the design of an application and so the methodology supported by our design platform allows modellers to either do a security analysis alone, or a performance analysis, or both. That is, the annotated versions of models which result from one run can be used again as inputs to Choreographer to perform a different type of analysis. The consequence of this is that a modeller using an established operational procedure to determine satisfactory levels of security (resp. performance) can use our design platform to do performance (resp. security) analysis alone. They are not forced to adopt both of the kinds of analysis which we offer if they do not need both, or already have a preferred way to do one of them.

The original contribution of this paper is to present a UML-based methodology for integrated security and performance analysis. The method is supported by a well-engineered tool and set on the formal foundation of dedicated process calculi with custom analysers. We describe the UML-based methodology which Choreographer supports and discuss the implementation of the Choreographer platform itself. We describe its use on a typical Web service creation problem: a Web-based micro-business. We believe that the Choreographer software tool could also be used for high-level analysis of other service-oriented architecture questions such as the assessment of service discovery protocols but we do not demonstrate this in the present paper.

Structure of this paper: The paper is structured as follows: the Choreographer analysis tool is presented in Section 2. The example application is a web-based micro business, described in Section 3. This is followed by a UML model and its associated performance and security models in Sections 4, 5 and 6. Related work and conclusions follow.

2 Choreographer

One feature of the methodology which we support with the Choreographer design platform tool is that modellers are able to express the models which are input to Choreographer in standard UML. The analysis process is initiated by invoking Choreographer on a UML project archive. The formal content of the UML model is stored in such an archive in an XML-based interchange representation (XMI). Software connectors termed *extractors* process the XMI representation of the input model and derive an analysable form of the model expressed in a process calculus. We use different process calculi for security and performance analysis: LySa [2] for the former and PEPA [3] for the latter.

Another key feature of the method is that the results of the analysis are reflected back as a modified version of the original UML model. The *reflectors* which do this are also available as software components which take the original UML project and the results of the analysers as inputs and write their results as complete UML projects in which the results of the analysis have been incorporated. The purpose of this is to ensure that the interpretation of the analysis results can be undertaken at the UML level and that the UML is not being used only as a model description language from which a process calculus representation is generated.Instead, reflecting the results back to the UML level centres the analysis on the UML description. We can picture this as in Figure 1.

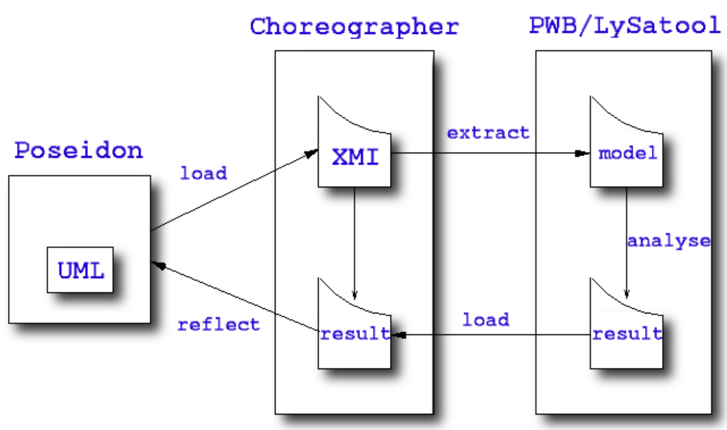


Fig. 1. Schematic of the Choreographer platform

The Choreographer platform is designed to support UML-centered development but is flexible enough to accommodate other modes of use in addition. These might simply be preferred by designers or developers who are using the platform or they might be needed to support a style of development favoured by the institution or software house which commissioned the development. Thus, a guiding principle of the design of Choreographer is that the processing of UML models should be made visible to the developer in order that the mapping between UML diagram elements and constructs of the process calculi beneath is transparent. This principle ensures that modellers have access to the representations which are needed to understand how their diagram elements are interpreted in the analysis process.

Exposing the functionality of the extractors and reflectors which map UML models to process calculi and analysis results back into the UML brings additional benefits for both implementors and users. We detail these in turn.

For the implementors the benefits of this design include having the functionality of the extractors open to inspection and enquiry. This greatly assisted with the problem of finding and correcting programming errors in the implementation of the extractors, with commensurate benefits to their implementation quality. The design of the analysis tools further supports the quality of the implementation of the extractors. The interface to each of the analysis tools is a formal language which is throughly checked by the tool before the analysis is carried out. Considering the performance analysis tools, models with deadlocks or missed synchronisations are rejected by the analysis tool, identifying errors in the input UML model or the implementation of the extractor which generated the process calculus representation.

For the users of the platform this has the benefit that it facilitates experimentation with models at the level of the process calculus, allowing a single UML model to give rise to a related collection of process calculus models. Here the UML level provides a much-needed layer of abstraction over the formal details. Such an abstraction would be useful in presenting a simplified—but consistent and accurate—view of the model to project managers or customers for whom the full complexity of the detailed process calculus representation would be a barrier to understanding.

Sophisticated users demand more from applications than raw functionality alone. The functionality must be presented in an appealing package if it is to attract a substantial user base and build a community of enthusiastic adopters. The issues here include engineering concerns which complement the scientific ones. Ignoring the engineering problems would be an unwise practise because the benefit to be derived from scientifically well-founded analysis will not be obtained if the analysers are not used. Within the engineering concerns are portability issues, user interface design and ease of installation. Modest problems in each of these areas would not deter the most enthusiastic adopter but could prevent an interested potential evaluator from trying the Choreographer platform in the first place.

To this end we worked to minimise engineering annoyances by building on portable, well-engineered implementation technology. Reasoning that we wanted to build an integrated development environment we researched a range of generic IDEs including Eclipse [4] and NetBeans [5]. Both of these build on the Java platform and provide generic user-interface and editor components, file system explorers, and other support for integrated development environments. Either could have been used for our purposes but we chose NetBeans because we judged that it had superior documentation about building plug-ins and applications.

NetBeans associates *data loaders* with file types and the functionality implemented in a data loader determines the processing which can be performed on the file contents. We implemented data loaders for each of the modelling languages used in our development method: UML (specifically, in its XMI format), PEPA and LySa.

By associated the necessary functionality with a language in this way, the Choreographer design platform guides the user through the modelling process. UML projects can be opened to obtain their XMI content. Extraction can be applied to the XMI representation to obtain a PEPA or LySa model. Performance analysis can be applied to PEPA models. Security analysis can be applied to LySa models. Results are reflected back into the UML representation. Choreographer guides the user through the process (extraction, then analysis, then reflection) and prevents “mode errors” such as trying to apply security analysis to a PEPA model, as could occur in a less structured modelling approach.

In terms of its appearance, the Choreographer platform follows the conventional design of an IDE, as seen in Figure 2. The main design area divides into an explorer on the left, an editor on the right, and a message console beneath these. The explorer provides a view onto the local file system which is structured in order to group related

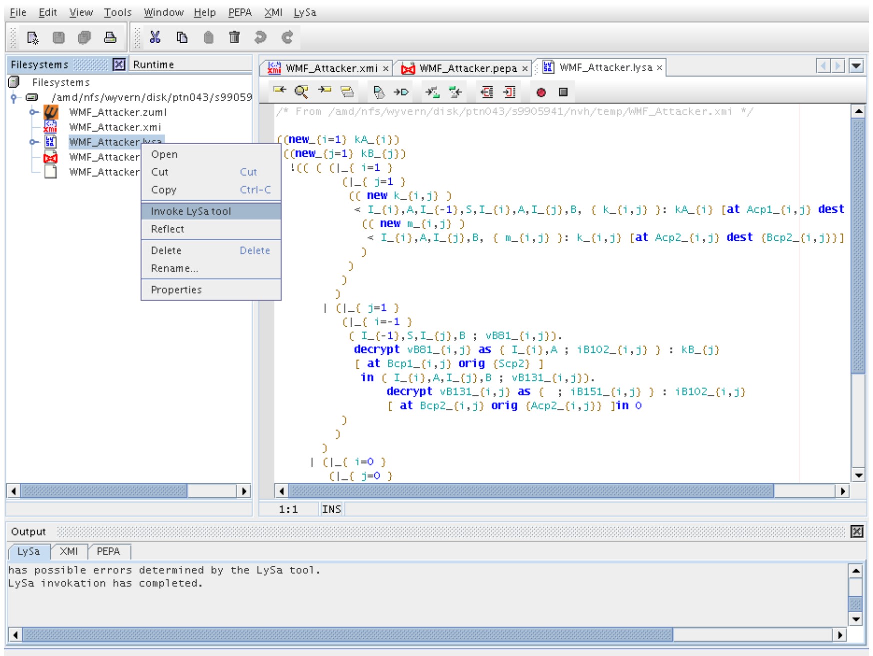


Fig. 2. The Choreographer user interface

documents into logical projects. The editor is language-aware with contextual modes: we have implemented editors for the process calculi which we use in the security and performance analysis process. The console is used to feed back to the user information about the progress of commands or analyses which have been launched from the application menus. Concise summaries of the analyses are printed into the console to allow information about the outcome to be obtained without having to initiate the reflection process and render the results in the Poseidon UML modelling tool.

When driving the performance and security analysis from the UML model exclusively, the editor pane in the Choreographer UI acts simply as a viewer, only used to display the extracted process calculus models. As we indicated earlier, an alternative is to work directly at the process calculus level. Working in this way, modellers either compose process calculus models directly, or modify ones which have been extracted from UML diagrams. Here we provide the expected level of formal language support with syntax highlighting, background lexical analysis and parsing, and context-sensitive diagnostic error reporting. Figure 3 shows Choreographer identifying an error in a PEPA model, calling the user's attention to the relevant part of the model and reporting the fault in the console below.

3 The web-based business system

The case study provided by our industrial partner is a business-to-business Web service to enable e-business based on a peer-to-peer authentication and communication

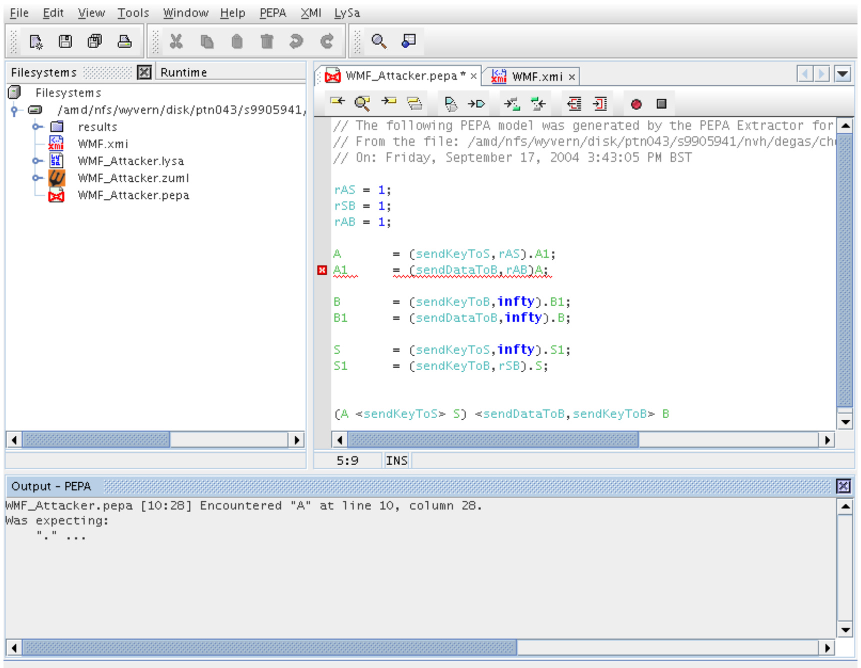


Fig. 3. Using Choreographer as a process calculus workbench

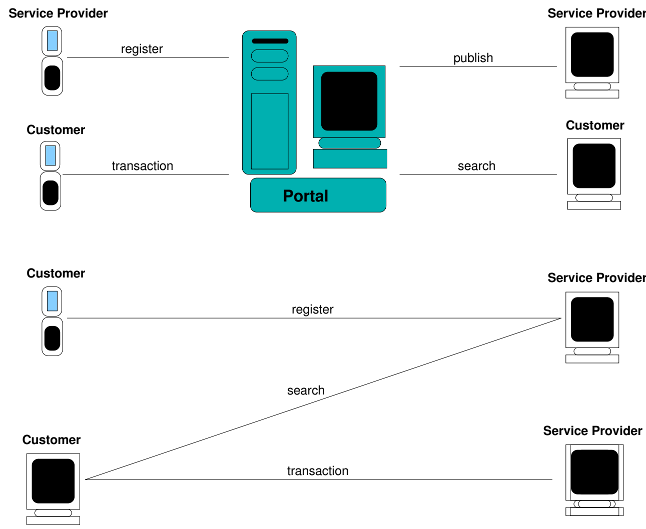


Fig. 4. Architecture of the web-based business system

paradigm. The objective of this system is to provide support to micro web-based businesses which do not themselves have the capability to develop proprietary solutions for e-business.

The service is accessible through both wired Internet connections and mobile devices using standard protocols such as the wireless application protocol. The system will present the various services offered by the service providers according to a coherent layout and will provide an interface for service access. While users should be able to process their transactions on a peer-to-peer basis, it is necessary to provide a central portal at which users register and can search for services. Registration and searching for services can be handled by UDDI.

The system naturally decomposes into three parts: the portal, service providers and customers (Figure 4). The upper part of Figure 4 describes that part of the functionality which involves the portal. The lower part concerns the peer-to-peer functionality.

The portal The portal enables remote data search and service navigation. Moreover it constitutes the interface between the customers and the service providers during the on-line business transactions. The e-business data management provides access to distributed products and services catalogues. The portal supports a significant number of concurrent sessions while providing end-to-end security of the transactions.

The service provider A new service provider joining the system first must register at the portal. A registered service provider can publish its services onto the portal dynamically. The list of its services can be accessed by any customer through the portal. Each provider will be able to modify its published services list by adding a new product; changing the characteristics of an existing one; or removing a service from the list. At any moment, a service provider can quit the system by unregistering from the portal.

The service provider can also handle transactions directly with customers who have registered at the service provider.

The customer Like the service providers, new customers have to register at the portal before being able to use its services. The registered customers are informed by the portal about available services, the newly published services, and the modified or removed ones. The user may perform on-line transactions via the portal to buy products he is interested in by selecting them from the list. The customers' order requests are then routed by the portal to the appropriate service provider. Alternatively, a customer can choose to communicate peer-to-peer with a chosen service provider after registering directly with this service provider.

Performance and security requirements There are many conceivable measures which one could apply to the quality of service which is to be ensured by the portal. These could include setting limits on the latency of the system, guaranteeing the responsiveness of the portal under typical operational conditions. Other measures could bring together performance and dependability elements to specify the required functional availability of the system. One of the reasons to target a well-supported performance modelling formalism such as PEPA for the performance analysis is that analysis tools are available for the computation of passage-time quantiles (namely the Imperial PEPA Compiler [6]) and for custom performability verification using model-checking (via the PEPA support provided by PRISM [7]). For the present study, however, we focus on performance evaluation based on the computation of throughput at the portal.

The security of the system is also crucial: Correct authentication at the portal and between service providers and buyers is required to prevent misuse by identity theft. The user data, in particular the transactions between service providers and customers, and also information associated to the operations to a user on the portal, should remain confidential, and integrity should be preserved. As all these data are transmitted wireless or via the web, strong security measures are required to meet these requirements.

4 UML model of the system

We turn now to our model of the above system. The performance model of the system consists of a collaboration between sequential object instances which undertake timed activities either individually, or in collaboration with other objects. Thus the UML diagram types which are used to describe this model are class diagrams (identifying the kinds of the objects in the system), state diagrams (detailing the behaviour of the objects) and collaboration diagrams (introducing an operational configuration of the system with named object instances collaborating on sets of activities).

4.1 Performance information

Performance analysis of the system is conducted via the generation and solution of a continuous-time Markov chain (CTMC) representation of the system, thus the durations

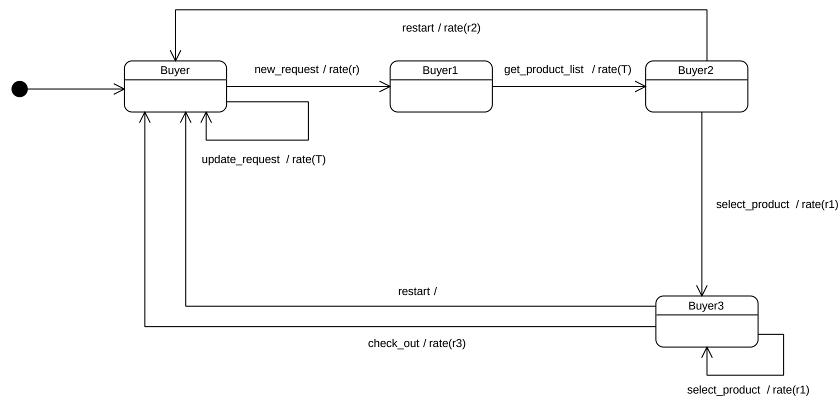


Fig. 5. State diagram of the Buyer in the Web-based micro-business model

of all of the activities in the system are quantified by providing the parameter to a negative exponential distribution.

Considering the arrival of new purchase requests into the system, we would represent this on the state machine diagram using an arc adornment of the form $new\_request/\mathrm{rate}(r)$. This form of adornment indicates that the performing object is sending $new\_request$ messages at rate r. This activity occurs in another form also. The recipient of the $new\_request$ message indicates that they are ready to receive such a message (but does not control how quickly they recur) with an arc adornment of the form $new\_request/\mathrm{rate}(\top)$. The $\top$ symbol indicates passive participation in this way. The effect is to mandate a synchronisation point between the two object instances with the asymmetry being used to model the caller/callee distinction between the object which sends the message and the object which receives it.

The state diagram which represents a buyer in the system is shown in Figure 5.

The diagram respects the UML meaning that object instances of this class are in one of the identified states of the system at all times and to transition from S_1 to S_2 there must be an arc which connects S_1 to S_2 leading in the direction from S_1 to S_2. Choices between different alternatives are indicated by having more than one outgoing arc from a state.

The relationship between the rate and the activity name in an arc adornment of the form $a/\mathrm{rate}(r)$ is that the average duration of an activity of type a is $1/r$. Thus, if that arc leads from state S_1 to S_2 then the average duration of that transition is $1/r$. The relative probabilities of the choices of successor states from a state can be obtained by renormalising the weights of the rate labels. Thus, if there are two outgoing arcs, one labelled $a/\mathrm{rate}(2r)$ and the other labelled $b/\mathrm{rate}(r)$ then the a-labelled arc will be followed twice as often as the b-labelled one (probability $2/3$ versus probability $1/3$, as expected).

Other components in the model are not much more complex than that of the buyers. Figure 6 shows that the model of the service providers in the system have common syn-

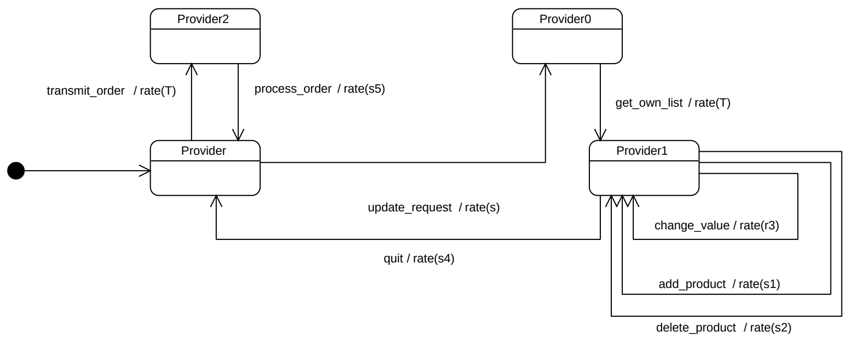


Fig. 6. State diagram of the Provider in the Web-based micro-business model

chronisation points with the buyers (reflecting exchanges which are not routed through the central portal in the system, for reasons of scalability). Where these synchronisation points occur, one of the interacting components specifies the rate of occurrence of the activity and the other passively co-operates with these activities.

The collaboration diagram in Figure 7 depicts an operational instance of the system with only two buyers ($b1$ and $b2$) and only two providers ($p1$ and $p2$).

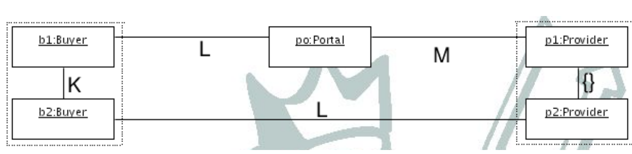


Fig. 7. Collaboration diagram for the Web-based micro-business model with annotations showing the grouping of objects of the same class

The collaboration diagrams which we use contain two important types of associations between object instances: associations between objects of the same class and associations between objects of different classes. Associations between objects of the same class have higher priority and should be thought of as grouping collections of correlated objects (even if these objects do not actually synchronise on any of their activities). We have suggested this grouping visually on the diagram by adding a dashed box around the grouped object instances. Thus we can think of the collaboration diagram as recording associations between the buyers in the system considered as a group, the providers in the system considered as a group, and the portal.

The associations in the system are labelled with the name of a set of activities. The end points of the association identify instances or groups of objects which are required to synchronise on the activities contained in the set, and not to synchronise on the activities which are not contained in the set.

The extraction of an analysable process calculus representation of this model is obtained by using Prowse's algorithm [8]. This is an efficient topological traversal of the collaboration diagram association graph, which performs alphabet inference on the associated component instances and builds a structured PEPA cooperation model of these on-the-fly. We extended the implementation of Prowse's algorithm in the PEPA extractor with a worklist-following phase utilising a stack of associations, and the concept of priority groups of object instances. These modifications mean that the algorithm retains the same low asymptotic complexity but is now able to process component replications with associations, which it was previously unable to do.

4.2 Security information

In the UML design, security relevant information is specified by the ForLySa profile [9], which provides the means to annotate class diagrams and sequence diagrams with security-specific data. More precisely, ForLysa allows us to specify cryptographic security protocols with two participants (A and B) who typically exchange a new session key. Such protocols use cryptographic concepts like cryptographic keys and nonces, which are provided by two classes in the ForLysa profile: the class *Msg* for messages and the class *Principal* for participants of the protocol. The class *Msg* has attributes holding the sender and receiver of the message and the encrypted and unencrypted payloads of the message; the latter are objects of appropriate classes, and these classes contain methods for encrypting and decrypting data. The class *Principal* contains attributes for the private/public keys or symmetric keys associated with a principal, and specifies methods for sending and checking of messages.

As an example, we show the UML design in Choreographer of the cryptographic security protocol described in Section 6, consisting of a class diagram and a sequence diagram. The class diagram, shown in Figure 8, specifies two principals A and B, as subclasses of *Principal*, which have attributes to hold the data generated or acquired during a run of the protocol.

The sequence diagram in Figure 9 describes the exchange of messages between A and B which defines the protocol. For each message, first the sender prepares and encrypts the content in method *premsg* by providing values for the attributes of a variable *out* of class *Msg*. When receiving a message, the recipient checks its contents (eg. correct addresses) with method *checkmsg*, then decrypts the encrypted parts with method *postmsg*. This assigns a value to the attribute which holds the decrypted content

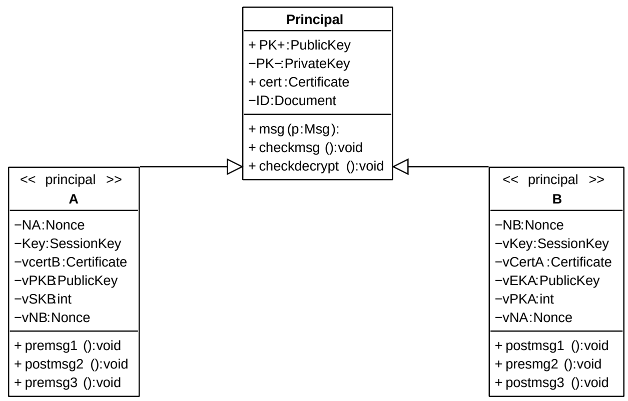


Fig. 8. The class diagram for the principals involved in secure transactions

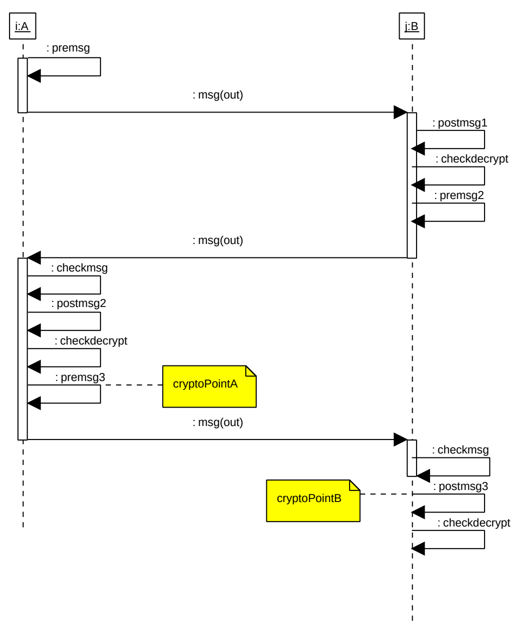


Fig. 9. The sequence diagram of the protocol for the principals involved in secure transactions

of the message. The decrypted part is then analysed in *checkdecrypt* where the receiver checks that the content has the required format. Figure 8 shows the call sequence for these methods, while the body of each method is specified by constraints which are not visible in the diagram.

5 Performance analysis

The performance analysis of the above UML project proceeds by extracting a performance model in Hillston's Performance Evaluation Process Algebra (PEPA) [3]. This extraction is performed automatically by the Choreographer design platform.

5.1 The PEPA model

The objects whose behaviour is specified by state diagrams in the UML model give rise to PEPA components in the process algebra model. The first component, *Portal*, models the behaviour of the interface between the service providers and the customers. The second component, *Provider,* models any provider registered in the system. The last component, *Buyer,* is used to model the behaviour of a customer. Note that in this model, we assume that both buyers and providers are already known to the system: they have already registered.

Component *Buyer* In an on-line transaction, the system user starts by sending a request to the portal about a specific product he is interested in—for example, books. This can be done by a simple click on the icon titled "Books" in the main pages of available products provided by the portal. This is modelled by action type *new_request*. The response of the portal is to send to the customer the catalogue or list of books available with all characteristics. We model this using action type *get_product_list*. Once the customer has the targeted list, he can select all the items he wants (action *select_product*) and then go to the check out (action *check_out*). This last step allows the buyer to place an order for selected items. At any moment the customer can change his mind and stop the process. This is modelled using action type *restart*. Note that action type *get_product_list* has an unspecified rate in component *Buyer* because the rate is defined by the portal which will send the list of products at his rhythm.

$$\begin{aligned}
Buyer \ &\stackrel{\mathit{def}}{=} (new_request, r).Buyer_1 + (update_request, \top).Buyer \\
Buyer_1 &\stackrel{\mathit{def}}{=} (get_product_list, \top).Buyer_2 \\
Buyer_2 &\stackrel{\mathit{def}}{=} (select_product, r_1).Buyer_3 + (restart, r_2).Buyer \\
Buyer_3 &\stackrel{\mathit{def}}{=} (select_product, r_1).Buyer_3 + (restart, r_2).Buyer \\
&\quad + (check_out, r_3).Buyer
\end{aligned}$$

Component *Provider* Once a service provider is registered, he may either send a request to the system to update the list of products or services he has published or receive an order from the portal. The former is modelled using action type *update_request* and the latter using action type *transmit_order*. In the first case, he will receive the list of services he owns (action *get_own_list*) and can then make all of the changes which he wants to using action types *add_product*, *delete_product* and *change_values*. Once he is finished with the updates he can leave the system (action type *quit*). In the second case, he will consider the customer order and do what is necessary to satisfy the request. This is modelled using action type *process_order*.

$$
\begin{aligned}
Provider &\stackrel{\text{def}}{=} (update\_request, s).Provider_0 + (transmit\_order, \top).Provider_2 \\
Provider_0 &\stackrel{\text{def}}{=} (get\_own\_list, \top).Provider_1 \\
Provider_1 &\stackrel{\text{def}}{=} (add\_product, s_1).Provider_1 + (delete\_product, s_2).Provider_1 \\
&\quad + (change\_values, s_3).Provider_1 + (quit, s_4).Provider \\
Provider_2 &\stackrel{\text{def}}{=} (process\_order, s_5).Provider
\end{aligned}
$$

Component *Portal* The portal manages both the buyers and the providers. All activities of component *Portal* are synchronizing activities, either with the buyers or the providers.

$$
\begin{aligned}
Portal &\stackrel{\text{def}}{=} (new\_request, \top).Portal_1 + (update\_request, \top).Portal_3 \\
&\quad + (select\_product, \top).Portal_1 + (restart, \top).Portal \\
&\quad + (check\_out, \top).Portal_2 + (get\_product\_list, v_1).Portal_1 \\
Portal_1 &\stackrel{\text{def}}{=} (get\_product\_list, v_1).Portal_1 + (select\_product, \top).Portal_1 \\
&\quad + (restart, \top).Portal + (check\_out, \top).Portal_2 \\
&\quad + (new\_request, \top).Portal_1 \\
Portal_2 &\stackrel{\text{def}}{=} (transmit\_order, v).Portal + (select\_product, \top).Portal_2 \\
&\quad + (restart, \top).Portal_2 + (check\_out, \top).Portal_2 \\
&\quad + (new\_request, \top).Portal_2 + (get\_product\_list, v_1).Portal_2 \\
Portal_3 &\stackrel{\text{def}}{=} (get\_list, v_2).Portal_3 + (add\_product, \top).Portal_3 \\
&\quad + (delete\_product, \top).Portal_3 + (change\_values, \top).Portal_3 \\
&\quad + (quit, \top).Portal
\end{aligned}
$$

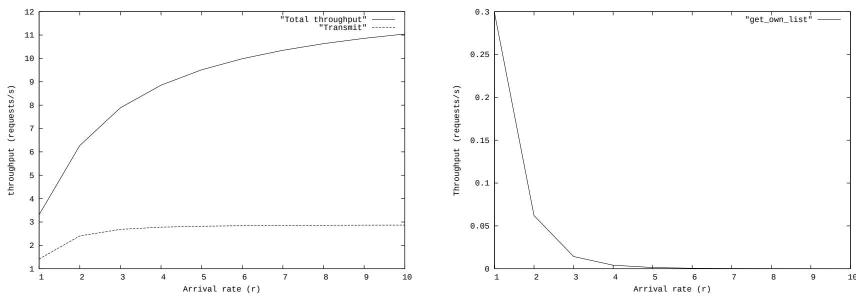
The complete system: The behaviour of the actors of the online system and their interactions between each other are captured by component $Web\_Business$ which is defined as follows:

$$
\begin{aligned}
&Web\_Business \stackrel{\text{def}}{=} \\
&\quad (Buyer \underset{K}{\bowtie} \ldots \underset{K}{\bowtie} Buyer) \underset{L}{\bowtie} \big((Provider || \ldots || Provider) \underset{M}{\bowtie} Portal\big)
\end{aligned}
$$

where the synchronising sets are defined as follows:

$$
\begin{aligned}
K &= \{update\_request\} \\
L &= \{new\_request, get\_product\_list, select\_product, restart, check\_out, \\
&\quad update\_request\} \\
M &= \{update\_request, get\_own\_list, transmit\_order, add\_product, \\
&\quad delete\_product, change\_values, quit\}
\end{aligned}
$$

Remark: The use of action *update_request* in component *Buyer* ensures that during the updates of a product list by its owner, the buyers do not have access to this list. As all components of the model must synchronise on *update_request*, it will not be enabled unless all occurrences of component *Buyer* are in their initial state.



(a) Total throughput (b) Throughput for provider requests

Fig. 10. Throughput computation

5.2 Numerical results

In this section we give an idea of the performance measures which we can compute in the context of such an application. We are mainly interested in the throughput of the portal. We consider a system composed of five buyers and one provider. This simple system allows us to retain intellectual control of the behaviour of the throughput in a system with a portal based architecture. All curves are plotted as a function of the arrival rate r of the requests of one buyer.

- Figure 10(a) depicts the total throughput of the portal in terms of buyer's requests to get a product list and to select a product from a list, and the provider's requests to get its own list. This figure also gives the throughput part related to the transmission of the orders to the provider. As we can see, the transmission of the buyer's orders is a very small part of the throughput of the system. This may be explained by the fact that the buyers spend the greater part of their time selecting products. Moreover, once an item is selected, a buyer may decide to abandon or restart. Thus not all buyers end up checking out with purchases.
- Figure 10(b) shows the behaviour of the part of the portal throughput related to the provider requests (*get_own_list*). Unlike what we have seen in Figure 10(a), this throughput decreases as the arrival rate increases. As we have more requests from the buyers, the portal spends more time dealing with these requests, and thus less time with the provider requests.

6 Security analysis

The security of a networked service depends heavily on the ability of users to send confidential messages via wireless or Internet connections, and to confirm the identity of the partner in their message exchange. Cryptographic techniques are usually used both to ensure the confidentiality of messages and for authentication.

But cryptography is not a magic wand to make everything all right. The main issue is that sending encrypted messages is only safe if only the authorized parties have the corresponding key. So data security becomes a key management problem [10], and the main task consists of designing an appropriate protocol for *authenticated key exchange*. Such a protocol allows two or more participants to exchange a cryptographic session key in such a way that the participants are assured that only the intended parties obtain the session key. Confidentiality and integrity of data is then guaranteed by encrypting all data with the session key. The main tool for providing proper authentication in such a key-exchange protocol is again cryptography, and hence an analysis tool must be able to deal with cryptographic concepts. Before describing the LySatool [2] used by Choreographer, we first discuss the security requirements of the web-based business system, and show the key exchange protocol chosen for the project. The protocol can be realised by the use of WS-Security, which provides all of the necessary mechanisms.

6.1 Security analysis for the web-based business system

In the case study, all communication should be encrypted to guarantee data confidentiality and integrity. This means that before starting a data exchange, a service provider and a customer or the portal and a user have to use a protocol for authenticated session key exchange.

For this protocol, there is a choice between using either symmetric cryptography or public key cryptography in a protocol for authenticated key exchange. When using symmetric key cryptography, the communication has to be conducted via a central server, and all users have to share initial symmetric keys with the server. The design goal of the project of providing peer-to-peer communication between service providers and customers would be violated if communication between users necessarily involved a central server. Moreover, initial distribution of secret symmetric keys is difficult to achieve in a practical way. Hence a protocol based on public key cryptography is used. In order to link a user identity U to a public key, it is essential to use certificates cert_U, e.g. X.509 certificates, which are signed by some trusted certification authority.

(1) A $\rightarrow$ B: A, cert_A
(2) B $\rightarrow$ A: $\{$B, NB$\}$:K_A^+, cert_B
(3) A $\rightarrow$ B: $\{$A, NB, $\mathrm{K}_{AB}\}$:K_B^+

The aim of the protocol is to provide authenticated key exchange between A and B, i.e. after the exchange both A and B are assured that only they know the new session key K_{AB}. More precisely, correct authentication is achieved by the protocol if A can be sure that message (3) can only be decrypted by B, while B knows that message (3) can only be sent by A.

6.2 LySa model of the protocol

The informal notation of the protocol used above leaves implicit a number of assumptions and does not completely describe actions such as decrypting with a certain key, comparing nonces, and checking certificates. Moreover it is crucial to specify the environment in which the protocol is executed, i.e. the actions which potential attackers can perform.

For a formal analysis, these assumptions have to be specified. LySa provides a format for this, which is essentially a process algebra, enriched by cryptographic notions such as encryption and decryption, symmetric keys, public and private keys, allowing it to model authenticated key exchange protocols. More precisely, LySa is based on the π-calculus. The main difference from the π-calculus and the Spi-calculus is that there are no channels: messages can be arbitrarily intercepted and redirected. Moreover, pattern matching is used to check that a message contains expected values (such as nonces), and to bind values to free variables. Each participant in the protocol (in our case A and B) is modelled by a separate process. Each message of the protocol corresponds to two actions: one performed by the sender who encrypts and sends the message, and one performed by the receiver who decrypts the message, checks the content, and might store parts of it.

As an example, consider message (3), sent from A to B. The LySa code for sending, which forms part of the process for A, is shown next. Sending of messages is denoted by $\langle \ldots \rangle$.

$$(\text{new} K_{AB})\langle A, B, \{|A, vNB, K_{AB}|\} : \mathrm{K}_B^+\rangle$$

The first argument in the $\langle \ldots \rangle$ expression denotes the sender (A), the second the recipient (B), and the rest is the content of the message. The content in this case consists of only one, encrypted, part. The terms are either names such as A, B, and K_{AB}, or variables such as vNB which has been bound to the value of NB when A received message (2). Sending message (3) is preceded by generating a new session key K_{AB} which nobody except A knows. This is modelled by restriction with the 'new' operator.

Input of a message is denoted by $(\ldots)$. We show the receiving action associated with (3), which is performed by process B:

$$(A, B; x).\text{decrypt } x \text{ as } \{|A, NB; vK|\} : \mathrm{K}_B^-$$

An incoming message is matched with an output, whereby the terms before the semicolon have to match while the variables after the semicolon are bound to values after successful matching. Accordingly, the first term denotes the sender and the second term denotes the recipient of the message. Encrypted terms are bound to a free variable and decrypted in the next step. Pattern matching is again applied to the content of an encrypted message. In the example, B only accepts the message if the first argument is A, and the second is the nonce NB which B has chosen for message (2). Note that B has to decide with which key to decrypt the message. For message (3), this is the private key K_B^-.

It does not suffice to code only one session between A and B, because attacks might require parallel sessions in which A or B (or both) participate. LySa offers the possibility to parameterise the protocol by n to code $n+2$ participants $I_{-1}, I_0, I_1, \ldots, I_n$, where $I_1, \ldots, I_n$ are the legitimate participants, I_{-1} is a server (not present in our example) and I_0 models an attacker who acts as legitimate participant. As described, the protocol consists of two processes: the process for A and the process for B. In the LySa model shown in Figure 11, every participant I_i can act either as A or B. Moreover, the replication operator ! indicates that any pair of participants perform an unlimited number of possibly concurrent sessions. The first line introduces the public/private keys of some certification authority, which are used to encrypt and verify certificates. In the second line, the public/private key pairs $PK+_i/PK-_i$ of all participants I_i are specified. Data (initially) not known to the attacker, like private keys, are restricted by using the 'new' operator. The attacker can read all messages and has hence access to all data contained in the unencrypted parts of messages, and if in possession of the corresponding key, can also decrypt messages. Since the public keys are assumed to be publicly known, we have to make sure that potential attackers have access to them. This is done by sending public keys in the clear in the last two lines of the protocol. The attacker built into the LySa model has the usual powers of the standard Dolev-Yao attacker [11], i.e. can use all information obtained from messages sent between participants to compose messages which can be sent to any participant. This means that a participant cannot be sure that the sender of a message is the one occurring in the first argument of the message, as the attacker has access to all names and can hence fake the names of sender and recipient in a message.

6.3 Security analysis with LySa

The analysis performed by the LySatool is to ask whether for multiple runs of the protocols between a number of participants, and in the presence of a standard (Dolev-Yao) network attacker, correct authentication is guranteed. The underlying technique is static analysis, more specifically the Succinct Solver Suite [12] provides the implementation of the solution procedures which are deployed to effect the analysis. LySa has been designed to verify correct authentication, and can also check confidentiality of data. The analysis of correct authentication is based on the use of assertions, which annotate the points in the protocol at which encryption and decryption takes place ('cryptopoints'). At an encryption point these assertions specify the destinations where it is believed that the complementary decryption can occur. At a decryption point the assertions specify the points where it is believed that the complementary encryption occurred.

For the key exchange protocol of the web-based business system, the LySa assertions specify that message (3) is correctly authenticated. More precisely, sending of message (3) is annotated with [at $a3_{i,j}$ dest $b3_{i,j}$] while receiving of message (3) has annotation [at $b3_{i,j}$ orig $a3_{i,j}$].

Hence, the assertions state correct (mutual) authentication of the communicating parties. The LySa tool checks whether an attacker is able to impersonate a legitimate participant and hence violate correct authentication. If the analysis shows that all assertions are correct in the presence of an attacker, we learn that the protocol guarantees correct authentication.

We have analysed the key exchange protocol for the web-based business system with LySa and shown that it provides authenticated key exchange. Moreover, we experimented with variants of the protocol and showed that omitting data from messages in

```
(new +- KCA)(
(new_{i=1} +- PK_{i})(
      /* process A */
(|_{i=1} |_{j=0\i} !(
        /* send (1) */
   <I_{i},I_{j},{|I_{i},PK+_{i}|}:KCA->.
        /* receive (2) */
   (I_{j},I_{i};v1_{i,j},vcertB_{i,j}).
   decrypt v1_{i,j} as {|I_{j};vNB_{i,j}|}:PK-_{i} in
   decrypt vcertB_{i,j} as {|I_{j};pB_{i,j}|}:KCA+ in
        /* send (3) */
   (new K_{i,j})(
   <I_{i},I_{j},{|I_{i},vNB_{i,j},K_{i,j}|}:pB_{i,j} [at a3_{i,j} dest {b3_{i,j}}]>.0
   )))
|
      /* process B */
(|_{j=1} |_{i=0\j} !(
       /* receive (1) */
   (I_{i},I_{j};vcertA_{i,j}).
   decrypt vcertA_{i,j} as {|I_{i};pA_{i,j}|}:KCA+ in
       /* send (2) */
   (new NB_{i,j})(
   <I_{j},I_{i},{|I_{j},NB_{i,j}|}:pA_{i,j},{|I_{j},PK+_{j}|}:KCA- >.
      /* receive (3) */
   (I_{i},I_{j};x4_{i,j}).
   decrypt x4_{i,j} as {|I_{i},NB_{i,j};vK_{i,j}|}:PK-_{j} [at b3_{i,j} orig {a3_{i,j}}]
   0)))
| |_{i=1} <PK+_{i}>.0
| <KCA+>.0
))
```

Fig. 11. LySa code for the security protocol in the web-based business system

the protocol makes it insecure. As an example, we show an attack which is possible when omitting the name A in message (3):

(1) A $\rightarrow$ B: A, $cert_A$
(2) B $\rightarrow$ A: $\{$B, NB$\}$:K_A^+, $cert_B$
(3) A $\rightarrow$ B: $\{$ NB, $K_{AB}\}$:K_B^+

After A has started a regular session with B, the attacker I starts a parallel session with B, and afterwards sends the response of B instead of the second message in the first session. Then the intruder intercepts the response of A in the first session and misuses it as message (3) in the second session.

(1) A $\rightarrow$ B: A, $cert_A$
(1') I $\rightarrow$ B: I, $cert_I$
(2') B $\rightarrow$ I: $\{$B, NB' $\}$:K_I^+
(2) I_B $\rightarrow$ A: $\{$B, NB' $\}$:K_A^+
(3) A $\rightarrow$ I_B: $\{$ NB', K $\}$:K_B^+
(3') I $\rightarrow$ B: $\{$ NB', K$\}$:K_B^+

The result is that K is the new session key for the session A thinks she is conducting with B as well as for the session between B and I. This means that I can intercept messages encrypted by A with the key K_{AB} and make B believe that the message comes from I.

7 Related work

Other authors have considered performance and security concerns in a unified framework. In [13] the security and performance demands of a secure electronic voting algorithm are considered. For the application of electronic voting, the requirement for a secure system with good performance and scalability is very compelling but the issue of integrated performance and security analysis also arises in small devices because of the cost of computing public and private key cryptographic routines. This is considered both by those working closely with present-day hardware [14] and by those working at the algorithmic modelling level [15, 16].

With regard to the performance analysis of UML models there are a range of significant prior works which have similarities with the performance-related part of our work. In many cases, these map UML diagrams of various kinds to other analysable representations including stochastic Petri nets [17, 18], layered queueing networks [19], generalised semi-Markov processes [20] and others. Some works are particularly noteworthy for their careful consideration of the role of the UML metamodel in the performance analysis process [21]. Our work has some similarities with the above, and many differences (different diagram types, different performance analysis technology). Two things are unique to our work here: an integrated technology for security analysis and the use of *reflectors* to reflect the results of the analysis back to the UML level.

Other methodologies based on UML have been defined in order to specify security aspects of designs. UMLsec by Jan Jürjens [22, 23] is a versatile profile that includes a wide range of high-level security concepts like secrecy, integrity, no-down-flow, fair exchange etc. and allows the user to specify hardware platforms such as LAN, smart card, Internet and others. It is however not possible to specify correct authentication, which is the main security requirement on the key exchange protocols which are part of the case studies that we have considered. As in the UML content processed by the LySa extractor, UMLsec protocols are specified by sequence diagrams, and the constraints used in the sequence diagrams are similar. However, the UML use supported by the LySa extractor provides a means to specify cryptopoints in sequence diagrams, which is an essential prerequisite for analysing correct authentication with LySa. Another modeling language for development of secure systems based on UML is SecureUML [24]. This work aims at role-based access control, while the LySatool focuses on authenticated key exchange.

8 Conclusions

We have presented a novel method for analysing security and performance questions about UML-described systems which follow a modern, open design pattern. The classes of behaviours understood within the system are described by class and state diagrams. The interactions between object instances of these classes are described using collaboration diagrams and sequence diagrams. The Choreographer design platform automatically processes descriptions of systems structured in this way, and packaged as a UML project. Process algebra representations of the formal content of the diagrams are extracted and passed to efficient analysers which check performance and security properties. The results of these analysers can be inspected directly or reflected back through the Choreographer design platform in order to present all of the analysis at the UML level.

The design and implementation of the Choreographer design platform leveraged the PEPA Workbench and LySatool analysers for the analysis effort and the NetBeans open IDE for building a custom graphical user interface for the application. The Choreographer platform has an open, extensible plug-in architecture which we are extending with other solution tools and model analysers.

We have applied the Choreographer platform to a range of small model examples and tests and a more substantial case study. We have found that the added richness of the interface has been appreciated as being an engineering improvement over the previous generation of analysis tools for process calculi. The representation of security and performance content is expressed in standard UML notation. As a helpful consequence of this design decision, Choreographer interoperates with standard UML tools such as Poseidon, without the need for additional diagram types or other extensions to the UML.

Through the use of the UML as an interface to the security and performance analysis process we hope that we have an accessible framework which could attract developers facing difficulties in engineering secure systems with high performance to consider formal analysis as a beneficial complement to their current design practices. There are many benefits to the use of formal modelling and analysis methods, not the least of which is the ability to display that due care and attention has been taken in the development of secure services which are to be used in business-to-business contexts.

It is not the case that an inexperienced modeller can use our system to compute any performance measure or test any security question that they wish without needing any understanding of the abstraction, modelling and mathematical analysis at work in performance prediction and estimation. However, we hope that we have gone some way to providing automated support for computing simple performance measures and classical security analyses. Using UML, we circumvent an unnecessary notational hurdle which could have been an impediment to the understanding and uptake of modern performance and security analysis technology.

Acknowledgements The work described in the present paper was undertaken while the authors were supported by the DEGAS (Design Environments for Global ApplicationS) project IST-2001-32072 funded by the FET Proactive Initiative on Global Computing. The Choreographer design platform incorporates the PEPA Workbench [25, 26] and the LySatool [27] as state-space generators and solvers. A modified version of the For-LySa extractor [9] is used as a component. We thank Matthew Prowse for helpful discussions on the extraction algorithm for PEPA. The Choreographer design platform is a Java application which has been successfully tested on Windows and Red Hat Linux systems. It is available for download from http://www.lfcs.ed.ac.uk/choreographer.

References

[1] Gentleware AG systems. Poseidon for UML web site, November 2004. http://www.gentleware.com/.

[2] C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H.R. Nielson. Automatic validation of protocol narration. In *Proc. of the 16th Computer Security Foundations Workshop (CSFW 2003)*, pages 126–140. IEEE Computer Security Press, 2003.

[3] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[4] Eclipse.org: open extensible IDE. Eclipse Foundation Web site, November 2004. http://www.eclipse.org/.

[5] NetBeans.org: open IDE. Web site of the NetBeans project, November 2004. http://www.netbeans.org/.

[6] J.T. Bradley, N.J. Dingle, S.T. Gilmore, and W.J. Knottenbelt. Derivation of passage-time densities in PEPA models using IPC: The Imperial PEPA Compiler. In G Kotsis, editor, *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pages 344–351, University of Central Florida, October 2003. IEEE Computer Society Press.

[7] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In J.-P. Katoen and P. Stevens, editors, *Proc. 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, volume 2280 of *LNCS*, pages 52–66. Springer, April 2002.

[8] C. Canevet, S. Gilmore, J. Hillston, M. Prowse, and P. Stevens. Performance modelling with UML and stochastic process algebras. *IEE Proceedings: Computers and Digital Techniques*, 150(2):107–120, March 2003.

[9] M. Buchholtz, C. Montangero, L. Perrone, and S. Semprini. For-LySa: UML for authentication analysis. In C. Priami and P. Quaglia, editors, *Proceedings of the second workshop on Global Computing*, volume 3267 of *Lecture Notes in Computer Science*, pages 92–105, Rovereto, Italy, 2004. Springer Verlag.

[10] Dieter Gollmann. *Computer Security*. Wiley, 1999.

[11] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 22(6):198–208, 1983.

[12] F. Nielson, H.R. Nielson, H. Sun, M. Buchholtz, R.R. Hansen, H. Pilegaard, and H. Seidl. The Succinct Solver suite. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, volume 2988 of *LNCS*, pages 251–265. Springer-Verlag, 2004.

[13] Nigel Thomas. Performability of a secure electronic voting algorithm. In Jeremy Bradley and William Knottenbelt, editors, *Proceedings of the First International Workshop on Practical Applications of Stochastic Modelling*, pages 81–93, London, September 2004. To appear in Electronic Notes in Theoretical Computer Science.

[14] Yusuke Matsuoka, Patrick Schaumont, Kris Tiri, and Ingrid Verbauwhede. Java cryptography on KVM and its performance and security optimization using hw/sw co-design techniques. In *2004 International Conference on Compilers, Architectures and Synthesis of Embedded Systems (CASES/04)*, pages 303–311, Washington, September 2004. ACM Press.

[15] C. Boyd and A. Mathuria. Key establishment protocols for secure mobile communications: a critical survey. *Computer Communications*, 23:575–587, 2000.

[16] Chiara Bodei, Mikael Buchholtz, Michele Curti, Pierpaolo Degano, Flemming Nielson, Hanne Riis Nielson, and Corrado Priami. On evaluating the performance of security protocols. In *Proceedings of the Second Workshop on Quantitative Aspects of Programming Languages*, 2003.

[17] J.P. López-Grao, J. Merseguer, and J. Campos. From UML activity diagrams to stochastic Petri nets: Application to software performance analysis. In *Proceedings of the Seventeenth International Symposium on Computer and Information Sciences*, pages 405–409, Orlando, Florida, October 2002. CRC Press.

[18] Juan Pablo López-Grao, José Merseguer, and Javier Campos. From UML activity diagrams to Stochastic Petri nets: application to software performance engineering. In *Proceedings of the fourth international Workshop on Software and Performance*, pages 25–36. ACM Press, 2004.

[19] D.C. Petriu and H. Shen. Applying the UML performance profile: Graph grammar-based derivation of LQN models from UML specifications. In *Proceedings of Tools'02*, number 2324 in LNCS, pages 159–177. Springer-Verlag, April 2002.

[20] C. Lindemann, A. Thümmler, A. Klemm, M. Lohmann, and O. P. Waldhorst. Performance analysis of time-enhanced UML diagrams based on stochastic processes. In Tucci [28], pages 25–34.

[21] S. Bernardi, S. Donatelli, and J. Merseguer. From UML sequence diagrams and statecharts to analysable Petri net models. In Tucci [28], pages 35–45.

[22] Jan Jürjens. UMLsec: Extending UML for secure systems development. In *5th Intl. Conference on the Unified Modeling Language (UML) 2000*, LNCS 2460, 2002.

[23] Jan Jürjens. *Secure Systems Development with UML*. Springer, 2004.

[24] Torsten Lodderstedt, Davin Basin, and Jürgen Doser. SecureUML: A UML-based modeling language for model-driven security. In *5th Intl. Conference on the Unified Modeling Language (UML)*, LNCS 2460, pages 426–441, 2002.

[25] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, number 794 in Lecture Notes in Computer Science, pages 353–368, Vienna, May 1994. Springer-Verlag.

[26] N.V. Haenel. *User Guide for the Java Edition of the PEPA Workbench—Tabasco release*. LFCS, Edinburgh, October 2003.

[27] Mikael Buchholtz. LySa — a process calculus. Web site hosted by Informatics and Mathematical Modelling at the Technical University of Denmark, April 2004. http://www.imm.dtu.dk/cs_LySa/.

[28] Salvatore Tucci, editor. *Proceedings of the Third International Workshop on Software and Performance (WOSP 2002)*. ACM Press, Rome, Italy, July 2002.