Performance tuning and cost discovery of mobile web-based applications

Matthias Book*, Volker Gruhn, Malte Hülder and André Köhler

Department of Computer Science
University of Leipzig
Klostergasse 3, 04109 Leipzig, Germany
E-mail: book@ebus.informatik.uni-leipzig.de
E-mail: gruhn@ebus.informatik.uni-leipzig.de
E-mail: huelder@ebus.informatik.uni-leipzig.de
E-mail: koehler@ebus.informatik.uni-leipzig.de
*Corresponding author

Abstract: When considering the addition of a mobile presentation channel to an existing web-based application, project managers should know how the mobile channel's characteristics will impact the user experience and the cost of using the application, even before development begins. The PETTICOAT (*Pe*rformance *T*uning and cos*t* d*isc*overy of m*o*bile web-based *A*pplica*t*ions) approach presented here provides decision-makers with indicators on the economical feasibility of mobile channel development. In a nutshell, it involves analysing interaction patterns on the existing stationary channel, identifying key business processes among them, measuring the time and data volume incurred in their execution, and then simulating how the same interaction patterns would run when subjected to the frame conditions of a mobile channel. As a result of the simulation, we then gain time and volume projections for those interaction patterns that allow us to estimate the costs incurred by executing certain business processes on different mobile channels.

Keywords: web engineering; mobile communications; cost estimation.

Reference to this paper should be made as follows: Book, M., Gruhn, V. Hülder, M. and Köhler, A. (2007) 'Performance tuning and cost discovery of mobile web-based applications', *Int. J. Web Engineering and Technology*, Vol. 3, No. 3, pp.254–270.

Biographical notes: Matthias Book is a doctoral candidate at the University of Leipzig. He holds a Diploma in Applied Computer Science from the University of Dortmund. His research interests are the development of the Dialog Flow Notation (DFN) for the specification of modular dialogue flows in web-based applications and the Dialog Control Framework (DCF) for the control of such dialogue flows on presentation channels such as desktop and mobile devices.

Volker Gruhn holds the Deutsche Telekom Chair of Applied Telematics and e-business at the University of Leipzig. Previously, he worked at the Fraunhofer Institute for Software and Systems Engineering (ISST), and later was a Professor at the University of Dortmund. He is a founder and Chairman of the supervisory board of the Dortmund-based adesso AG. Gruhn is author and co-author of about 120 national and international publications. His research interests include agile and model-driven methods for the development of distributed and mobile e-business applications.

Malte Hülder is a doctoral candidate at the University of Leipzig. He holds a Diploma in Computer Science from the University of Dortmund, where he also previously worked as a Research Associate. His research interests are in the area of mobile applications, especially hybrid connectivity for always-online applications.

André Köhler is a doctoral candidate at the University of Leipzig. He holds a Diploma in Business Information Systems from the University of Leipzig. His research focus is on Mobile Process Landscaping, a method for the systematic analysis of distributed process landscapes and the identification of potential for process optimisations that can be accomplished through the use of mobile technologies.

1 Introduction

As thin-client applications, web-based applications have the advantage of independence from the user and his preferred device. Only the existence of a browser and a suitable network connection are needed. Thus, web-based applications seem to be convenient for mobile use. But in hands-on trials of such scenarios, the response time of the application is often notably worse compared to its use in a LAN environment. Furthermore, the communication costs are hard to predict. An organisation that plans to provide mobile access to its existing web-based applications for a large group of mobile workers needs detailed information about response times and estimated cost of the application in a mobile environment *before* investing any effort in building it. Therefore, the expected response time, as well as the expected cost of the application on different mobile networks needs to be quantified at an early stage. With PETTICOAT (*Pe*rformance *T*uning and cos*t* d*isc*overy of m*o*bile web-based *A*pplica*t*ions), we present a method that can be used for this purpose.

The PETTICOAT method can be used by software developers as well as software project managers. After compiling all necessary information, a tool calculates indicators that reveal the application's response time and communication costs in the mobile environment. This way, decisions on the development of a mobile channel for an application can be based on quantitative arguments. If the application is classified as not immediately suitable for mobile use, decision-makers can use the detailed results to consider whether it is reasonable to address particular deficits in the application's design revealed by the simulation. This optimisation can be conducted for single features or the whole application.

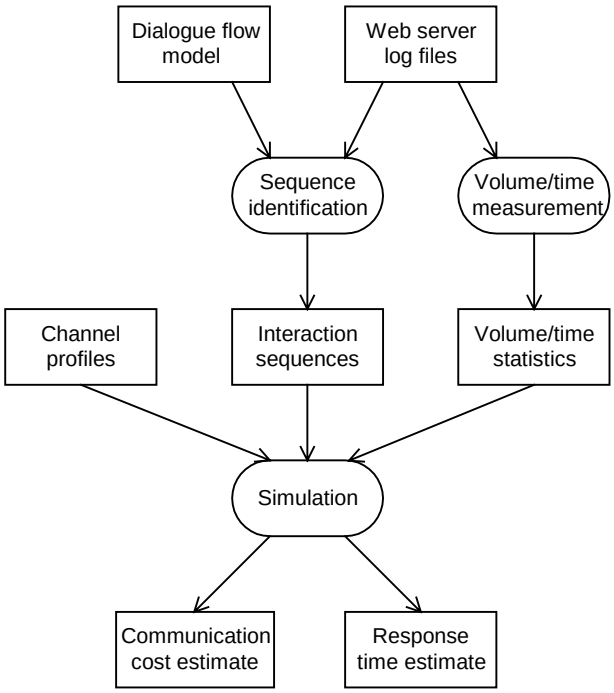
In this paper, we describe how the PETTICOAT method was employed in a case study that we performed in cooperation with an insurance company. The following section presents each step of the method in detail. Using examples from the case study, we show how to model the application structure as a dialogue flow (Section 2.1), identify typical interaction sequences within the application (Section 2.2), measure the time and data volume in the existing application (Section 2.3), specify the mobile channels' characteristics (Section 2.4), simulate the application's interaction sequences on different

mobile channels and evaluate the response times and cost implications of the observed time and data volume (Section 2.5). After an overview of the related work in Section 3, Section 4 provides a short summary and outlines ongoing and future work in this area.

2 The PETTICOAT method

The PETTICOAT method provides decision makers with indicators on the economical feasibility of mobile channel development. In a nutshell, it involves identifying interaction sequences in a dialogue flow model of the existing application, measuring the time and data volume incurred in their execution (either by analysing web server log files or observing real-time traffic), and then simulating how the same interaction sequences would perform when subjected to the frame conditions of a mobile channel. As a result of the simulation, we gain time and volume projections for the interaction sequences that allow us to estimate the cost incurred by working with the application on different mobile channels (Figure 1).

Figure 1 The PETTICOAT method



The following subsections present these steps in more detail and illustrate them with excerpts from a case study we performed for an insurance company. In that project, we applied the PETTICOAT method to the prototype of a new web-based offer management system in order to estimate the cost that will be incurred each month by insurance agents accessing the system over mobile networks such as GSM, GPRS and UMTS.

2.1 Modelling the dialogue flow

As a basis for our analysis, we need a model of the application's complete dialogue structure. We use the Dialog Flow Notation (DFN) (Book and Gruhn, 2004) for this purpose. This graphical notation models an application's dialogue flow as a directed graph of states that are connected by transitions. We call the transitions 'events' and the states 'dialogue elements', distinguishing 'masks' (web pages rendered on the client) and 'actions' (business logic executed on the server). Events can carry parameters that transport business data, such as form input.

2.1.1 Manual dialogue graph specification

By building dialogue graphs from masks, actions and events, the developer can specify all possible user interactions with the application. To increase the expressive power of the specification, parts of a dialogue graph can be encapsulated in 'dialogue modules' that can be reused in different contexts within the same application by nesting them into the dialogue flow at arbitrary levels. This allows the developer to build complex dialogue structures that closely mirror the users' mental model of the complex business processes supported by large-scale web applications.

While the notation may seem suitable only for static websites with a finite number of pages at first sight, it can model database-driven websites with dynamically generated pages just as well, if we assume that one mask represents a class of similar page instances that are all embedded in the same navigational structure (for example, in an online shop, we do not need to model individual masks for each product, but just one *Product Details* mask that can show the details of any product). Note, however, that a different modelling approach would be needed for more complex GUIs rendered using AJAX, applets or other technologies, since those rely on a different interaction paradigm and thus produce different communication patterns.

As an example, Figure 2 shows the dialogue graph of the offer management system analysed in the case study. Since we were looking at a rather simple prototype, the model does not make use of the DFN's dialogue modularisation capabilities and comprises only seven dialogue masks connected through a number of actions that implement various business operations, an exemplary selection of which is shown in Table 1. Field staff users enter the application through the *initialise system* action *(0)*, which leads to the *Search Transaction Form* mask *(A)* where they can look up, create or edit transactions. Using the other masks and actions, transactions can be associated with insurance agents, insurance holders and policy offers.

In order to use these graphical specifications as input for the following steps, they can be automatically translated into the XML-based Dialog Flow Specification Language (DFSL) (Book and Gruhn, 2004). However, we do not show this straightforward conversion step here for the sake of brevity.

Figure 2 Dialogue graph of the offer management system

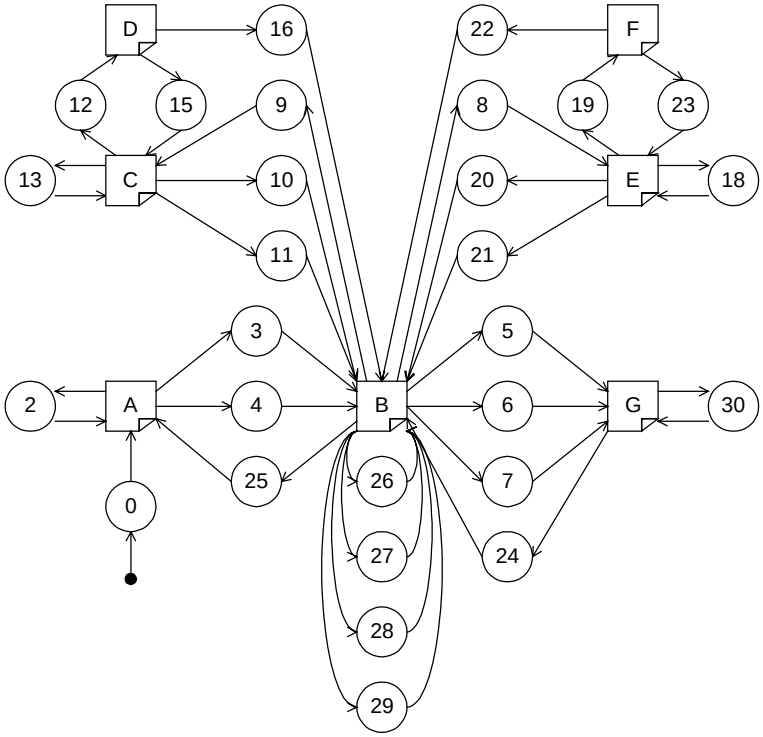


Table 1 Masks and actions of the offer management system (excerpt)

Mask identification	*Content*
A	Search Transaction Form
B	Edit Transaction Form
C	Associate Agent Form
D	Create Agent Form
E	Associate Insurance Holder Form
F	Create Insurance Holder Form
G	Edit Offer Form
Action identification	*Function*
0	Initialise system
2	Search transactions
4	Prepare transaction for editing
6	Prepare offer for editing
…	…
26	Expand transaction elements
27	Collapse transaction elements
28	Load documents
29	Browse transaction elements
30	Process offer modifications

2.1.2 Automatic dialogue graph reconstruction

Since the PETTICOAT method is typically employed to assess existing applications that were built without using the DFN, modelling their dialogue graphs *a posteriori* may be a tedious task. However, this work can be supported by the automatic reconstruction of the dialogue graph from information found in web server access log files. For each request, these log files record (if configured properly) the URL of the requested page, the URL of the previous (referring) page, and information that can be used to associate requests with user sessions. In principle, we could iterate over the log file entries and populate the dialogue graph with a mask for each requested page and an event for each referral from one page to another within the same user session.

In practice, this approach does not work quite as smoothly because the information in the log files is not unambigous: For one thing, if no unique session identifier is included in the log entries, we need to rely on heuristics, such as a combination of the user agent name and the IP address found in the log entry in order to identify sessions, which may not be entirely accurate. As a second and much harder challenge, the URL recorded in the log file may not clearly identify the page that is ultimately presented to the user, since it usually points to some processing logic (*e.g.*, a user authenticity check) that may deliver different pages based on the processing outcome (*e.g.*, a 'login successful' or 'login failed' page).

In order to identify these pages unambiguously, we need to capture additional information in the log files. This can be accomplished either by inserting tiny transparent images with unique URL parameters into the delivered pages, which inject additional requests into the web server log files containing the desired information, or by identifying the selected page at a suitable central point just before it is delivered (*e.g.*, within the application's dialogue control logic, or in a filter intercepting all responses going out to clients) and recording the information in a separate log file. The choice between these methods depends on the architecture of the application – generally, developers should aim to keep to a minimum the required level of invasion into the existing logic in order to minimise the development effort and risk of introducing errors.

One might argue that even when the challenges of unambiguous page and session identification are addressed, the automatically reconstructed dialogue flow model may not be complete, *i.e.*, it may not comprise all possibilities for navigation that are offered by the application, if those were not recorded in the web server log files (*e.g.*, if a certain link was never used, the respective request was never logged). However, for the purpose of the PETTICOAT method, this is not a problem since we are only interested in those paths that users actually traverse, anyway. It is also likely that the auto-generated dialogue flow model will have to be manually cleaned up a bit before being used for further processing, since it may contain redundant events or 'loose ends' that are introduced by prematurely terminated sessions, discontinuities in the referral chain caused by backtracking, or semantic considerations that are not apparent from the raw log entries. However, we believe that this manual vetting will require less effort than specifying the complete dialogue graph of a complex web application from scratch.

2.2 Identifying interaction sequences

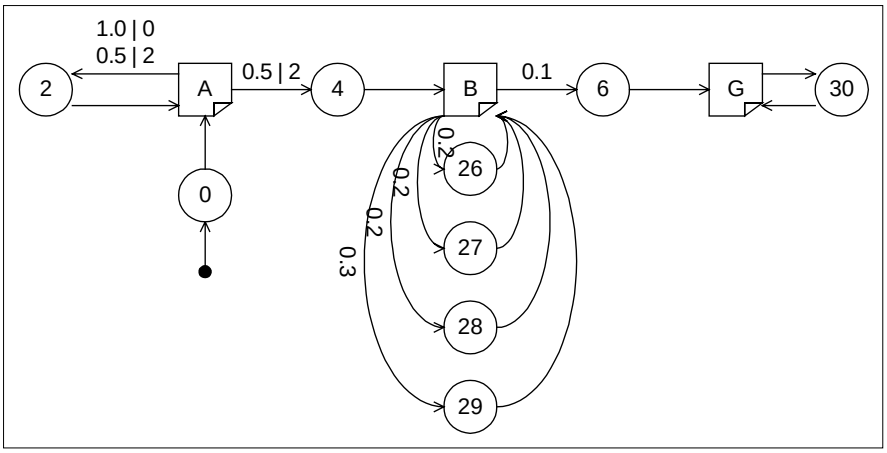
The dialogue graph of an application specifies all possible ways of interaction that the user interface allows. Since the same business process may be accomplished in a number of similar, but still different ways, there will typically be some more and some less frequently traversed paths through the dialogue graph (called 'interaction sequences' from now on). To arrive at a representative cost projection for the business processes performed with the application, we therefore need to analyse the actual interaction sequences that occur in the application. By identifying the sequences that the users traverse most frequently, we can later weigh the cost they incurred, accordingly.

In the case study, we identified and analysed 15 interaction sequences (*i.e.*, subsets of the whole dialogue graph), based on the tasks that users can perform in the offer management system. In web-based systems, every possible way through the application could be defined as one interaction sequence. The restriction to the 15 most important interaction sequences was requested by the company that developed the mobile application we evaluated. Decisive factors for the restriction were the importance of a sequence for completion of a business process and the number of recurrences in a given period. To extend the scope of our analysis, we could also have looked for further interaction sequences in the web server log files that were performed frequently by users, even though they did not lead to completion of a task – those sequences would then hint at usability problems in the application. By analysing those unsuccessful sequences, too, we could determine how much (unnecessary) cost they incur.

As an example, Figure 3 shows the sequence for finding a transaction, browsing its elements and editing the associated offer. The events in this sequence are annotated with probabilities to reflect the different possibilities of executing this business process. Since a user's interaction steps are not isolated from each other, but depend on the history of his interactions, these probabilities are conditional: In the notation, we first note the probability of a user following this event, and then (after a vertical bar) note which action the user must have executed before as a condition for this probability. For example, from mask *A*, there is a 1.0 probability that the user will execute action *2* under the condition that he executed action *0* before, but a 0.5 probability that he will execute action *2* if he already executed that action before. In other words, if the user just entered the application, he will definitely use the search feature, but if he already searched for a transaction, there is only a 50% probability that he will use the search feature again. Rather, there is also a 50% probability that he will proceed to edit the transaction he found (denoted by the 0.5 | 2 probability for the event leading to action *4*). For events without annotation, the implicit probability of traversal is 100%, regardless of the previously executed action.

Obviously, the events' probabilities may depend on a longer history than just the last executed action. For those cases, our interaction model allows the specification of preconditions comprising multiple previous actions. We are currently investigating the level of history that needs to be incorporated into this model in order to achieve sufficiently accurate approximations of the users' behaviour.

Figure 3 Interaction sequence for finding and editing an offer associated with a transaction



In our case study, the probabilities were estimated based on practical considerations. Alternatively, a more realistic probability model can be obtained by evaluating user tracking information that is routinely collected in web server log files (Pitkow and Pirolli, 1999). This can be accomplished through an algorithm similar to the one used to reconstruct the application's dialogue graph in the previous step: First, we manually need to define the masks and events that a certain business process comprises (*e.g.*, *A*, *B*, *G* and their connecting actions and events in the example in Figure 3). This step cannot be automated since it requires knowledge of the masks' and events' semantics, which cannot be deduced from the log files. After this, we can automatically iterate over the log files, looking for patterns that mirror the possible paths through the defined interaction sequence. In the process, we count the frequency of each observed variation of the interaction sequence (*e.g.*, *0-A-2-A-4-B* versus *0-A-2-A-2-A-4-B*), and calculate the event probabilities and preconditions from these findings (for the sake of brevity and focus on the overall PETTICOAT approach, we will not go into further detail on the details and challenges of this algorithm here).

While it is helpful to visualise the resulting interaction sequences graphically as in Figure 3 during the conceptualisation phase of the study, they ultimately need to be converted to a machine-readable format in order to be processed by the simulation tool. We use a variation of the DFSL for this purpose. The resulting sequence specification also contains estimates on how often each sequence will be executed by each user each month, which will be used towards the end of the simulation in order to calculate the approximate monthly cost of executing all sequences.

2.3 Measuring data volume and time

As mentioned in the introduction, the two main factors influencing the cost of interaction with an application over a mobile channel are the time spent online and the data volume transmitted. To project these metrics for mobile channels, we measure them on the existing stationary channel and then input them into the simulation.

There are a few challenges in the details of this measurement process, however. Most importantly, for the volume measurement, we need to distinguish between static and dynamic content. While static content (such as images) always incurs the same volume (apart from caching effects, which can be accounted for in the simulation), dynamic content (such as search result pages) can produce a different volume for each request. To obtain accurate estimates, we need to deduce a probability distribution or an average value from the accumulated volume data. Also, web server log files only log the net volume of the content, but not any overhead introduced on lower levels of the protocol stack that nevertheless does count for billing purposes. This overhead can either be ascertained by observing the data flow directly on a sufficiently low protocol level instead of relying on server log files, or by factoring it into the simulation in close accordance with the respective protocol specifications.

In our case study, we used a simple HTTP traffic listener to obtain the necessary data. The characteristics of each web page, image, *etc.*, were described in an XML-based format where each of those 'web elements' is represented by a `WebElement` tag that contains tags for its various attributes: Tags starting with `Request` or `Response`, for example, contain the data volume incurred for the request and the response of the web element in bytes, depending on whether the web server configuration allows HTTP compression or not. The `Inlines` tag contains references to web elements such as images included in a page. For each of these, we can specify the offset of their include point on the page (*i.e.*, the number of bytes of the parent web element that need to be loaded before the inline web element is requested by the browser). As an example, the document excerpt in Figure 4 shows the description of the *Search Transaction Form* mask in our case study.

Figure 4 Web element specification for *Search Transaction Form* mask

```
<WebElement>
     <ID>A</ID>
     <Desc>Search Transaction Form</Desc>
     <ElementType>text/html</ElementType>
     <Cache>0</Cache>
     <RequestUncomp>750</RequestUncomp>
     <RequestComp>500</RequestComp>
     <ResponseUncomp>9000</ResponseUncomp>
     <ResponseComp>3000</ResponseComp>
     <Inlines>
       <Inline>
         <WebElementID>100</WebElementID>
         <OffsetComp>1200</OffsetComp>
         <OffsetUncomp>5000</OffsetUncomp>
       </Inline>
     </Inlines>
</WebElement>
```

To measure the time it takes to complete an interaction sequence, a number of contributing factors need to be considered. The total time a user spends online is the sum of user activity (*e.g.*, filling in forms), upstream and downstream transmission time, channel latency and server processing time. To accurately distinguish all these contributing factors, we would need synchronised timing on both the server and the client, which is hard to guarantee. Fortunately, however, only the user and server activity matter for the subsequent simulation, since the observed transmission time and latency already depend on the stationary channel that we measured on. We can thus deduct them from the overall time during the simulation based on our knowledge of the stationary channel characteristics and volume transmitted. This way, we are left with the user and server activity time, to which we can add the newly calculated transmission time and latency based on the mobile channel's characteristics. Alternatively, if we are measuring on a channel with very high bandwidth (*e.g.*, an Ethernet), we can just measure the time difference between sending a response to the client and receiving the next request on the server, and regard this interval as the user activity time, since the included transmission time (typically a fraction of a second) is much shorter than the user activity (typically quite a few seconds) in comparison and can thus be neglected.

In preparation for the following steps of the PETTICOAT approach, the measured timings are specified for each action, *i.e.*, each transition between masks, in an XML-based format similar to the one shown in Figure 4.

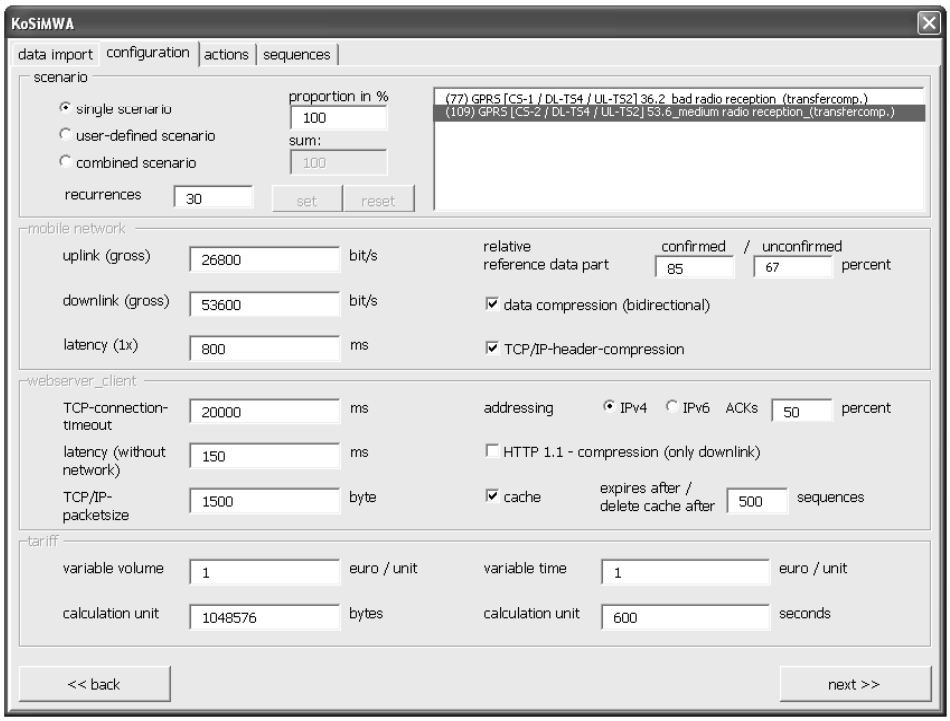
2.4 Defining channel profiles

Besides the description of the application's interaction patterns, mask and action characteristics, we still need a detailed specification of the target (usually wireless) network environment, since different wireless networks have different characteristics regarding bandwidth, latency, pricing, *etc.* We define these characteristics in XML-based 'channel profiles' for each network that shall be considered in the simulation. In our case study, we defined 16 channels, including different compression variants for GSM, HSCSD, GPRS and UMTS networks. The tool also considers the effects of fluctuating signal strength. Each profile contains the gross uplink and downlink bandwidth in bit/s, network latency for wired and wireless networks, and the average ratio of gross data and content data. In addition to these data describing the physical network characteristics, a channel profile contains several attributes for packet and compression characteristics describing HTTP, TCP and IP protocols. Furthermore, the network provider's rates for volume-based and time-based billing are contained in the profile description.

Thus, different assumptions about the quality of reception, use of compression algorithms, certain protocol features, or provider's rates result in different channel profiles. Channel profiles are stored in XML documents and can be edited by the user in the simulation tool, as illustrated by the definition of a GPRS 53.6 channel profile with data compression under the assumption of medium reception in Figure 5.

The rates in the case study were based on pricing plans of a German telecommunications provider. Channel and protocol characteristics were defined based on the respective technical specifications, and a number of profile variants for different connection quality levels were defined for each channel, based on estimates and provider information. Note that the accuracy of the simulation results could be improved further by defining channel profiles based on parameter measurements in the target environment, rather than drawing them from the specifications.

Figure 5 Specification of channel characteristics in the simulation tool



2.5 *Simulation of interaction sequences on different channels*

In order to perform the simulation, our tool requires the XML documents produced in the previous steps as input, *i.e.*, the application profile that contains the web elements and their volume data, the actions and their timing data, and the sequences with their probability and frequency data; and also the channel profiles containing the bandwidth, latency and pricing characteristics.

Using this input, the simulation tool works in three steps that will be described in detail in the following sections:

1 The simulator begins each interaction sequence at its entry point. Taking into account the branching probabilities, it then simulates the time it takes to load each mask in the sequence, considering the inline element offsets, which incur latency and traffic that delay the completion of the mask. This step already yields insights into usability problems that may be caused by unacceptably high response times.

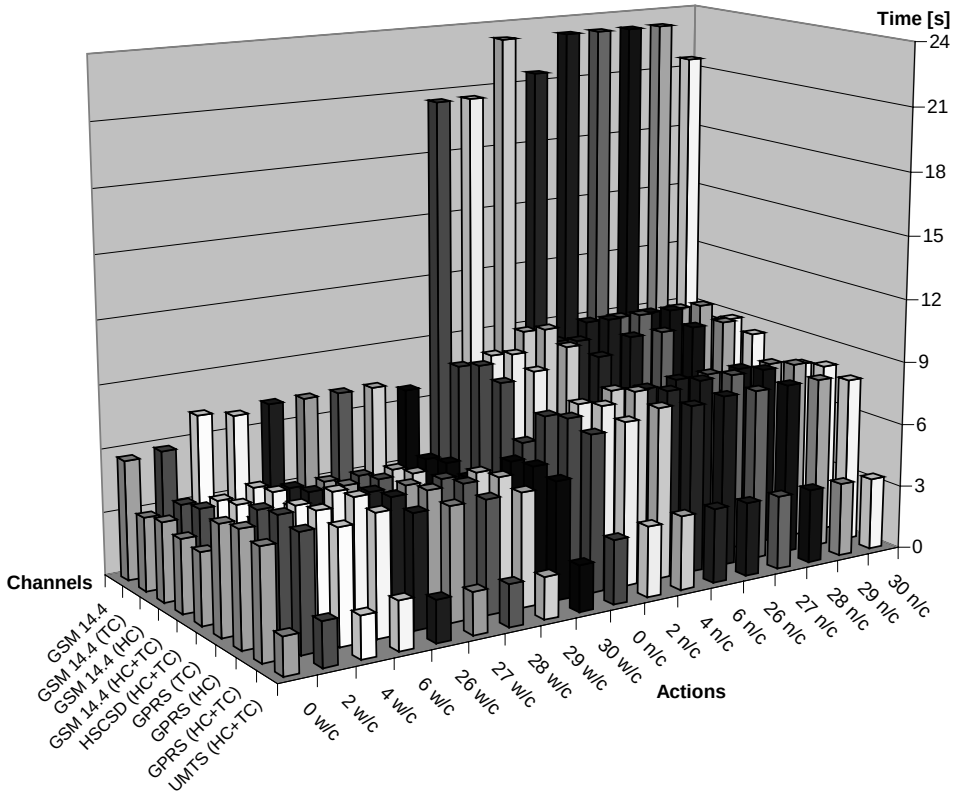
2 The results for each interaction step of a sequence are accumulated, taking the user's idle time between interactions into account.

3 The results for each interaction sequence are multiplied by its estimated frequency per user and month. Summing up the results finally yields the projected communication costs of the application per user and month.

2.5.1 Simulation of interaction steps

The simulation results for the response times of the actions (*i.e.*, the transitions between the masks) in the interaction sequence shown in Figure 3 on a selection of channels are given in Figure 6. The diagram clearly shows that the use of a client-side cache (actions marked 'with cache (w/c)') significantly reduces the response time in contrast to executing the same sequence without a cache (marked 'no cache (n/c)'). However, it may still be relevant to investigate an application's response time without a cache since not all mobile platforms provide sufficient cache memory.

Figure 6 Simulation results for response times



From Figure 6, we can also determine that only the UMTS channel with enabled browser cache supports answering times of less than three seconds for this application and thus provides adequate usability (if we follow Shneiderman's (2002) rule that response times for simple actions should not exceed one second, while the maximum response time for standard actions is four seconds).

Figure 6 also indicates that in our case study, the response times on the GPRS channel are longer than those on the GSM or HSCSD channel (using the identical compression and caching mechanism). This may come as a surprise, as GPRS may provide three to four times the bandwidth of a GSM 14.4 channel. On the other hand, GPRS has a network latency of about two seconds, so any request is delayed by about two seconds

before the transmission of the requested data actually begins. By that time, the requested data would already have reached the recipient on the GSM channel. Only if the cache is deactivated, the GPRS channel can make up for the latency with its higher bandwidth.

Other, more complex timing constraints than the above-mentioned three-second response time rule are also conceivable. For example, we could define the constraint that mask *n* has to be loaded within *t* seconds after leaving mask *m*. The results gained in the simulation may then indicate which masks are responsible for failing the constraints and need to be optimised. If no redesign seems feasible, the application cannot be used on the simulated channel with the specified constraints. For example, in our case study, the results on the GPRS channel indicate that a redesign should particularly focus on embedding inline elements at the top of a mask, so that requests for these elements can be sent earlier by the browser, and the network latency is mitigated by the remainder of the page still being loaded in parallel.

2.5.2 Simulation of interaction sequences

In the second step, the simulation tool sums up the results for the individual steps in a sequence gained in the first step. It also adds the user's estimated idle time to simulate how long a user works with a mask on average before the next mask is requested. This way, the tool determines how long it typically takes to execute a whole interaction sequence on a channel (taking the different probabilities for the sequence variants into account), and how many bytes are transferred in the process. Using the providers' rates specified in the channel profiles, the tool can then calculate the cost of performing each interaction sequence on each channel.

In Table 2, an excerpt of the results gained for a selection of channels and sequences from our case study is given. For each of the available channels (GSM, HSCSD, GPRS and UMTS), four simulations were carried out using no compression, HTTP compression (by the web server), transfer compression (by the carrier), and both HTTP and transfer compression. Sequence 4 denotes the process of a user creating a new policy offer, which has to be associated with an existing insurance agent and a new insurance holder. Sequence 11 represents the process of creating a new policy offer by copying an existing one. Finally, sequence 12 contains the results for finding and editing a policy offer, as shown in Figures 3 and 6. For each channel/sequence combination, the table contains the time taken to execute the whole sequence, the total amount of kilobytes transferred in the process, and the cost incurred under a time- and volume-based pricing plan on the respective channel. Since volume-based billing is not available for GSM and HSCSD channels, the respective fields remain blank.

The results indicate that the use of data compression reduces the data volume to roughly a third of the uncompressed volume, resulting in lower transmission times. It is important to note, however, that when using transfer compression, the carrier will charge for the uncompressed data volume. HTTP compression thus seems to be the better choice for volume-based pricing plans, as only the reduced data volume is billed. This effect can be observed, *e.g.*, when comparing the results for scenario 11 (with and without cache) on the GPRS channel with transfer versus HTTP compression. A volume-based plan also allows for more flexibility regarding idle times, since longer client-side activities before requesting the next mask are not billed. On the other hand, transfer compression seems to be the best choice for time-based plans, because it yields the shortest transfer times

resulting in lower charges. Combining both transfer and HTTP compression may combine their advantages, but because of a greater overhead and slightly longer execution time on the web server, this combination may not yield the lowest cost regarding time and/or volume.

Table 2 Simulation results for performing interaction sequences on different channels (excerpt)

Sequence	*Cache?*	*Results*	*GSM 14.4*	*HSCSD (HTTP Comp. + Transfer Comp.)*	*GPRS (Transfer Comp.)*	*GPRS (HTTP Comp.)*	*GPRS (HTTP Comp. + Transfer Comp.)*	*UMTS (HTTP Comp. + Transfer Comp.)*
4	w/c	Time [s]	81,4	62,9	75,2	76,1	74,6	52,0
		Volume [kB]	61	22	62	22	23	22
		Charge (t) [€]	0,41	0,28	0,13	0,13	0,12	0,09
		Charge (vol) [€]	–	–	0,06	0,02	0,02	0,02
4	n/c	Time [s]	206,2	76,5	89,2	90,6	88,6	58,7
		Volume [kB]	292	109	293	109	109	109
		Charge (t) [€]	1,03	0,34	0,15	0,15	0,15	0,10
		Charge (vol) [€]	–	–	0,29	0,11	0,11	0,11
11	w/c	Time [s]	185,4	129,0	156,2	160,5	154,4	110,2
		Volume [kB]	143	41	137	42	41	43
		Charge (t) [€]	0,93	0,58	0,26	0,27	0,26	0,18
		Charge (vol) [€]	–	–	0,13	0,04	0,04	0,04
11	n/c	Time [s]	420,8	165,3	191,4	203,4	199,5	124,7
		Volume [kB]	580	220	566	221	222	212
		Charge (t) [€]	2,10	0,74	0,32	0,34	0,33	0,21
		Charge (vol) [€]	–	–	0,55	0,22	0,22	0,21
12	w/c	Time [s]	210,1	149,1	191,0	184,7	182,9	129,9
		Volume [kB]	150	43	152	45	45	45
		Charge (t) [€]	1,05	0,66	0,32	0,31	0,31	0,22
		Charge (vol) [€]	–	–	0,15	0,04	0,04	0,04
12	n/c	Time [s]	467,5	193,8	222,4	223,8	217,3	150,8
		Volume [kB]	621	236	620	226	223	230
		Charge (t) [€]	2,34	0,87	0,37	0,37	0,36	0,25
		Charge (vol) [€]	–	–	0,61	0,22	0,22	0,23

2.5.3 Simulation of monthly usage

In the final simulation step, the tool uses the results gained so far to project the total cost that will be incurred when one user works with all interaction sequences in the application over the course of one month. This enables project managers to estimate the total communication costs that can be expected on all channels, and decide if the addition of a mobile channel will pay off.

For our case study, the final results indicated that a UMTS channel with combined transfer and HTTP compression and a volume-based pricing plan is the best option. This scenario would incur an estimated monthly cost of €55.11 per user. A volume-based plan on a GPRS channel with transfer and HTTP compression costs only €54.94 per user and month, but exhibits worse usability owing to the longer response times brought about by high network latency, as Figure 6 illustrated. Since UMTS is currently not available all over the country, GPRS can still be recommended as a suitable backup solution with limited usability. The time-based plans for the HSCSD and GSM channel would result in monthly costs of €298.35 and €421.19 per user, respectively, with both using only transfer compression, since the combination of transfer and HTTP compression would be even more expensive in total.

3 Related work

The PETTICOAT approach employs a number of techniques from the web usage and web data mining fields, as described by Cooley (2003), Srivastava *et al.* (2000) and Kosala and Blockeel (2000). Dutta *et al.* (2001) show how frequent and thus critical user paths can be identified in e-commerce applications. The authors provide a model of the user behaviour in the form of session graphs and conduct analyses regarding the most frequently used user paths as well as critical edge sequences. This technique could be quite useful for our approach, because the identification of the most frequently used subset of all possible user paths in the application model is needed.

Furthermore, there are many approaches for web log analysis aimed at classifying user paths (*e.g.*, Spiliopoulou, 2000; Berkhin *et al.*, 2001; Kim *et al.*, 2004; Heer and Chi, 2002; Chi *et al.*, 2000; Gillenson *et al.*, 2000). Especially, the identification of long sequences described by Pitkow and Pirolli (1999) seems to be an important topic for the PETTICOAT concept. The identification of actually chosen user paths versus all possible user paths in the application model is needed in order to obtain meaningful results from the following simulation. In this context, the work of Mao *et al.* (2001) is of specific interest. They present a notion for a cluster-based online monitoring system for web traffic. The target-oriented analysis of web traffic is a task to be solved within the PETTICOAT approach.

As PETTICOAT particularly addresses the analysis of dynamic web applications instead of static web pages, the analysis of web traffic is even more difficult. This problem is addressed, *e.g.*, by Berendt and Spiliopoulou (2000), which deals with dynamic web content generation and website analysis.

Other approaches to improving the performance of web-based applications have focused on using thin clients to transmit just the image of the application (see *e.g.*, Lai *et al.*, 2004). The findings of this work are of relevance to the deduction of consequences (application design, bandwidth restriction) based on the simulation results. In this

context, Bent *et al.* (2004) and Krishnamurthy and Wills (2000) report interesting results from an analysis of large websites regarding performance, cache and cookie issues. These results could be used for the creation of a package of measures in order to modify the analysed website regarding performance issues in the mobile environment.

4 Conclusion and future work

In this paper, we have shown a method for assessing the response times and communication costs of adding mobile channels to an existing web-based application. As illustrated by the case study, the results of the simulation indicate if an existing application can be accessed efficiently on certain mobile channels, and provide clues on how the application may have to be optimised for shorter response times. The simulation also provides an estimate of the cost of using the application on various mobile channels, which is a valuable factor in deciding if the introduction of a mobile channel will pay off for an organisation in the future.

In our ongoing work, we currently focus on the automated analysis of web applications to simplify the initial steps of the PETTICOAT method. This includes deriving the dialogue flow model and the probabilities and frequencies of typical interaction sequences from the data contained in web server log files, rather than modelling them manually. Further research will comprise refinements of the probabilistic model for the interaction sequences and a more detailed specification of mobile channel characteristics and billing schemes in order to increase the accuracy of the estimates, and thus the quality experienced by users when accessing web-based applications through mobile channels.

Acknowledgements

The authors would like to thank Andreas Kriegel for his development of the simulation tool, Adrian Bensch for his work on interaction sequence identification and Matthias Pätzold for his work on dialogue graph reconstruction. The Chair of Applied Telematics/e-Business is endowed by Deutsche Telekom AG.

References

Bent, L., Rabinovich, M., Voelker, G.M. and Xiao, Z. (2004) 'Characterization of a large Web site population with implications for content delivery', *WWW '04: Proceedings of the 13th International Conference on the World Wide Web*, ACM Press, pp.522–533.

Berendt, B. and Spiliopoulou, M. (2000) 'Analysis of navigation behaviour in Web sites integrating multiple information systems', *The VLDB Journal*, Vol. 9, No. 1, pp.56–75.

Berkhin, P., Beche, J.D. and Randall, D.J. (2001) 'Interactive path analysis of Web site traffic', *KDD '01: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp.414–419.

Book, M. and Gruhn, V. (2004) 'Modeling Web-based dialog flows for automatic dialog control', *19th IEEE International Conference on Automated Software Engineering (ASE 2004)*, IEEE Computer Society Press, pp.100–109.

Chi, E.H., Pirolli, P. and Pitkow, J. (2000) 'The scent of a site: a system for analyzing and predicting information scent, usage, and usability of a web site', *CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press, pp.161–168.

Cooley, R. (2003) 'The use of web structure and content to identify subjectively interesting web usage patterns', *ACM Transactions on Internet Technology*, Vol. 3, No. 2, pp.93–116.

Dutta, K., VanderMeer, D., Datta, A. and Ramamritham, K. (2001) 'Discovering critical edge sequences in e-commerce catalogs', *EC '01: Proceedings of the 3rd ACM Conference on Electronic Commerce*, ACM Press, pp.65–74.

Gillenson, M., Sherrell, D.L. and da Chen, L. (2000) 'A taxonomy of web site traversal patterns and structures', *Commun. AIS*, Vol. 3, No. 4, p.5.

Heer, J. and Chi, E.H. (2002) 'Separating the swarm: categorization methods for user sessions on the web', *CHI '02: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press, pp.243–250.

Kim, D-H., Atluri, V., Bieber, M., Adam, N. and Yesha, Y. (2004) 'A clickstream-based collaborative filtering personalization model: towards a better performance', *WIDM '04: Proceedings of the 6th Annual ACM International Workshop on Web Information and Data Management*, ACM Press, pp.88–95.

Kosala, R. and Blockeel, H. (2000) 'Web mining research: a survey', *SIGKDD Explorations*, Vol. 2, No. 1, pp.1–15.

Krishnamurthy, B. and Wills, C.E. (2000) 'Analyzing factors that influence end-to-end web performance', *Proceedings of the 9th International World Wide Web Conference on Computer Networks: The International Journal of Computer and Telecommunications Networking,* North-Holland Publishing Co., pp.17–32.

Lai, A.M., Nieh, J., Bohra, B., Nandikonda, V., Surana, A.P. and Varshneya, S. (2004) 'Improving web browsing performance on wireless PDAs using thin-client computing', *WWW '04: Proceedings of the 13th International Conference on the World Wide Web*, ACM Press, pp.143–154.

Mao, Y., Chen, K., Wang, D. and Zheng, W. (2001) 'Cluster-based online monitoring system of web traffic', *WIDM '01: Proceedings of the 3rd International Workshop on Web Information and Data Management*, ACM Press, pp.47–53.

Pitkow, J. and Pirolli, P. (1999) 'Mining longest repeating subsequences to predict World Wide Web surfing', *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*.

Shneiderman, B. (2002) *User Interface Design,* mitp-Verlag.

Spiliopoulou, M. (2000) 'Web usage mining for web site evaluation', *Commun. ACM*, Vol. 43, No. 8, pp.127–134.

Srivastava, J., Cooley, R., Mukund, D. and Pang-Ning, T. (2000) 'Web usage mining: discovery and applications of usage patterns from web data', *SIGKDD Explorations*, Vol. 2, No. 1, pp.12–23.