RESOURCE SUBSTITUTION FOR THE REALIZATION OF MOBILE INFORMATION SYSTEMS

Hagen Höpfner and Christian Bunse
International University in Germany
Campus 3; 76646 Bruchsal; Germany

Keywords: Resource Awareness, Software Engineering, Adaptivity, Mobile Information Systems.

Abstract: Recent advances in wireless technology have led to mobile computing, a new dimension in data communication and processing. Market observers predict an emerging market with millions of mobile users carrying small, battery-powered terminals equipped with wireless connection, and as a result, the way people use information resources is predicted. However,the realization of mobile information systems (mIS) is affected by the users need to handle complex data sets as well as the restrictions of used devices and networks. Hence, software engineering has to bridge the gab between both worlds, and thus, has to balance given resources. Extensive wireless data transmissions, that is expensive, slow, and energy intensive can - for example - be reduced if mobile clients cache received data locally. In this short paper we discuss, which and how resources are substitutable in order to enable more complex, more reliable and more efficient mIS. Therefore, we analyze the resources used for data management with mobile devices and show how they can be considered by software development approaches in order to implement mIS.

1 INTRODUCTION

Due to the recent advent of wireless technologies evolve, the mobile systems and especially mobile information systems (mIS) have an impact on numerous facets of our daily lives and the way business is conducted. According to (Gurun and Krintz, 2003) mIS are aimed at providing important data in real time to assist decision makers, will exert great influence on communications between businesses and their customers, and will or already has transformed the way we live our lives.

In detail, mobile information systems are often client server information systems with mobile clients.[1] Mobile clients (e.g., laptops, PDAs, smart phones, or even Java enabled mobile phones) connect to static and powerful information system servers. Since such devices are, on the one hand, widely used but have, on the other hand, various limitations, this paper focus on the usage of lightweight mobile devices as information systems clients.

[1]In this short paper we do not address pure mobile peer-to-peer approaches that are discussed in literature but do not play a bigger role in real life at the moment.

When realizing information systems that use or support mobile devices not only functionality but also resources play a major role. Important resources for mobile information systems (mIS) are: memory /energy consumption, wireless communication, CPU performance etc. These resources cannot be viewed in isolation since they are often inter-related with each other (e.g., extensive wireless communication has a negative impact onto energy consumption). Thus, balancing strategies are required. These strategies can be subsumed by the term *resource substitution*. A good example is the implementation of data caches on cell phones to avoid mass data transmissions. However, this requires available memory. Thus systems, with limited memory and CPU power might use means of communication for handling complex data. But, wireless communication is energy intensive and results in a reducing the availability of the device. Thus, resource substitution requires careful thinking.

This paper presents some initial ideas on how support static resource substitution at development time as well as dynamic resource substitution at runtime can be achieved. The idea of substituting resources is not new. It was already discussed in (Rudenko et al.,

1998) with regard to the energy consumption and on a more abstract level in (Buchmann, 2002). However, addressing resource substitution as part of the software development process in order to support static and dynamic resource substitution has not been properly addressed yet.

The remainder of the paper is structured as follows: Section 2 gives a brief overview of typical resource characteristics. Section 3 shows how these resources are used in mobile (client server) information systems. Section 4 discusses their substitutability. Section 5.1 includes issues regarding resource substitution at development time. Section 5.2 presents our ideas on how to dynamically substitute resources at runtime. Section 6 summarizes the paper, provides some conclusions as well as an outlook on future research.

2 RESOURCES

Mobile devices have to be mobile! This implies that they have to be portable, almost independent from power sockets and wired network connectors. However, users want to use them for purposes comparable to tasks supported by classical desktop computers. Beside communicating via voice or SMS, users want to access (distributed) information (König-Ries et al., 2007). In order to implement software for mobile devices that fits the users needs one has to consider the resources provided by the mobile device and the environment infrastructure. In the following we describe such resources:

CPU: There are various CPUs for mobile devices on the marked, e.g. XScale or other processors based on the ARM architecture. Furthermore, there exist mobile devices that have dual CPUs like Nokias E90. (Marwedel, 2007)

In fact, the performance of mobile CPUs increases. Samsungs SCH-i760, that will be released in June 2007, will use a 400 MHz Samsung SC32442 CPU connected over an 100 MHz internal system bus.

Memory: Storing data on mobile devices is different from storing data on classic computers. In contrast to big hard-discs (secondary storage) or DVD-drives (ternary storage) mobile devices typically use flash memory. Some PalmOS devices use this media as main memory, too. Others, like HP iPAQ hx4700 series Pocket PC, separate main memory (SDRAM) and secondary storage (flash). Nearly all current mobile devices allow the use of additional storage media in form of memory cards (e.g. CompactFlash or SmartMedia). However, these specifications describe only the interfaces and size of the media. In addition, there are also “real” hard discs (e.g., IBMs Microdrive) that can be plugged into a CompactFlash Typ-II slot. Such devices can store up to 4GB of data.

Energy supply: Mobile devices are battery powered. Their up-time mainly depends on the intensity of usage (e.g., a Motorola Razor V3 has a standby time of more than a week. Using it for surfing the web or chatting via GPRS the battery has drained out after less than one day). Current research tries to solve this problem by improving hardware aspects (e.g., using fuel cells). Although, this seems to be a step into the right direction there are limitations. For example Nokia decided in 2005 to stop all research in this direction due to legal issues.

Wireless networks: Data communication in mobile information systems is wireless. Today, a mobile device has basically network access almost everywhere. Wireless local-area networks are hotspoted networks with a reasonable speed. GSM-networks (and extensions like GPRS) are slower but widely available. UMTS-infrastructure has been installed all over Europe but wide-area coverage is far from existing. In addition to its limited availability, wireless communication is energy intensive (see above).

3 RESOURCE USAGE IN MOBILE INFORMATION SYSTEMS

As mentioned in Section 1 mobile information systems typically are client-server information-systems using mobile clients. Therefore, they have to query data from a server, compute and display them on the mobile client, and synchronize changes (if existing) with the server. The first phase can be considered to be a synchronization without client-side updates. Hence, we are able to analyze the resources needed for performing the remaining tasks:

If a client starts operation, it requests the required data from the server. This can be done via replication, where the client stores the received data locally, or by using simple query-answer approaches comparable to those of web browsing. Of course, latter can be supported by caching mechanisms. The amount of required memory depends on whether data has to be stored locally. Full replication, guaranteeing data availability, requires more memory than caching received data using a cache replacement strategy. Browser-like behavior does not require specific memory for storing results, although in general each computation requires memory.

Regarding the use of network facilities resource-replication is one of the most efficient strategies. Since all required data might be available on the mobile device, only the delta needs to be transmitted. Caches might replace certain data, thus, if data is requested later, it has to be retransmitted. Without replication and caching, data has to be transmitted at any time it is required. Therefore, this approach is network, and indirectly energy, intensive.

However, if data must not be locally stored, CPU usage can be minimal since local data must not be filtered or stored locally. The replication of reused date case requires that all data has to be available on the mobile device and that requests are posted to the local replication system. This increases CPU usage. Concerning caching and especially semantic caching complex algorithms are needed (Ren and Dunham, 2003). Since only fractions of requested data might be available, the software system has (a) to find reusable parts, (b) request missing data, and (c) combine them parts. Unfortunately, some of these algorithms are known to be NP-hard and therefore CPU intensive.

Storing data is energy intensive but requesting data wirelessly requires even more energy. Hence, browsing approaches require more energy than approaches that store data locally. However, they are not always the “best” solution since the energy consumption of memory and CPU has to be considered as well. Experiments (Marwedel, 2007) show, that even complex calculations consume less energy than accessing the memory. Hence, caching seams to be more energy efficient than replication. Nevertheless, missing data must be retransmitted, which might countermand the savings by caching.

In addition there are techniques that query and cache data automatically with regard to user needs or context. However, such hoarding or pre-fetching approaches have the nearly the same resource consumptions as replication and query-answer with caching. The only difference being an increase in CPU usage in order to do computations according to user needs and context awareness.

4 SUBSTITUTABILITY OF RESOURCES

In the Section 3 we have discussed, that different techniques for handling data in mobile information-systems have different resource consumptions. Table 1 gives an general overview about possible substitutions that holds for all distributed systems, and thus also for client/server information systems with mobile clients. According to this table the resource communication might be replaced by CPU or memory resources (e.g., do computations and store data locally). Unfortunately, table 1 does not include energy as a separate resource due to its orthogonal nature being affected by other resources. However, due ever increasing functionality of mobile devices, limited battery capacities and environmental protection, energy consumption has to be addressed too. In the following we discuss in-depth the substitution possibilities regarding mobile information system (mIS) with respect to energy consumption.

4.1 Cpu Vs. Communication

High CPU load can be substituted by data communication. CPU-intensive calculations might be outsourced to a server. Concerning replication or non-caching approaches this is not relevant if not warranted by the data-processing. However, there are scenarios where data processing is complex; e.g. (Ion et al., 2007) prohibiting mobile phones to handle them.

As mentioned earlier caching is a CPU-intensive approach. (Höpfner, 2004) presents an approach for migrating the task of finding reusable data in the cache to the server. In addition, (Höpfner, 2006) analyzes server site cache invalidation. Although, both approaches require complex software operations on the server, they drastically reduce client CPU load. However, replacing communication by CPU load is only applicable in modern systems given a low number of data changes concerning server and client. For example mobile navigation systems do not need data-communication since all required data is already locally available. However, most application scenarios are of a more dynamic nature. Reducing communication by replication techniques then has an impact on the resource memory.

4.2 Cpu Vs. Memory

From the database point of view the substitution of CPU load by memory access and vice versa is the question of materializing views in a database system. Hence, the mobile client has to decide on storing temporary results. Solving this problem is not easy (see also Section 2) out of the following reasons: If the mobile device uses only flash memory as primary (main memory) and secondary storage, computed data is materialized anyway. If the mobile device uses a (slow) microdrive materialization may cause serious performance problems.

Table 1: Substitutable resources (Buchmann, 2002).

substitute	CPU	Communication	Memory
CPU	-	Migration of computations to the server	Materialization and re-usage of temporary results
Communication	Local execution of calculations	-	Local data storage
Memory	Data compression and compact data structures	Data management on the server only	-

In order to save CPU cycles on the mobile client one can also think about using additional or improved index structures to ease data localization. In the context of caching approaches it makes sense to generalize cache-descriptions (Höpfner and Schallehn, 2004), or to introduce a hierarchical access path in order to reduce the complexity of the required algorithms. Furthermore, standard compression approaches might be used for saving memory.

4.3 Memory Vs. Communication

As mentioned in Section 3 replication and caching provide mechanisms for reducing the amount of communication. However, full replication reduces communication, whereas the efficiency of caching depends on the re-usability of cached data (Caracaş et al., 2007). So, the benefit of caching depends not only on the memory but also on the CPU. In general one can say, data that is available on a mobile device must not be transmitted. But, redundant data might be outdated. Hence, replication and caching need synchronization mechanisms that connect to a server or receive updates via a broadcast channel.

In summary, the substitution of memory by communication is obvious but not practicable since wireless data-transmission is expensive. Interestingly, there are approaches following this substitution strategy (e.g., the internet message access protocol (IMAP) allows to read email without prior download).

4.4 Energy Vs. CPU, Memory, Communication

Energy consumption is typically correlated with hardware properties. However, there are strategic issues that can be addressed from the software point of view:

- Energy consumption associated with wireless data-transmission is not negligible (Feeney and Nilsson, 2001).
- Storing data locally requires less energy than wireless transmissions depending on the data-size, storage time, etc..
- CPU usage needs comparatively less energy than memory storage (Marwedel, 2007).

With regard to mobile information systems this means: Storage- or transmission-intensive approaches (e.g., replication and query-response) have a higher energy consumption than caching. However, this assumption can not be proven now but is a focus of our current research.

5 SOFTWARE ADAPTABILITY AND STATIC RESOURCE SUBSTITUTION

The previous sections talked about resource substitution strategies in order to reduce the energy consumption of a mIS. Systematic development of such systems requires sound engineering techniques that support system adaptation. In general, we have to examine when such an adaptation occurs and how the adaptation is performed. As for the former, adaptation can occur at development, compile/linking, load time, and runtime. For the latter, adaptation can be accomplished by a change in algorithms/behavior, by a change of parameters or by a change of the software composition. In the following we discuss static and dynamic resource substitution in more detail.

5.1 Static Resource Substitution

Interestingly, research concerning the impact of software onto energy-consumption puts a specific focus at the later phases of software development. In detail these are the algorithmic level (i.e., using the most efficient algorithms and data structures) (Pakdeepaiboonpol and Kittitornkun, 2006), high-level source code transformations (Falk and Marwedel, 2005), compiler optimizations (Chen et al., 2006), code-compression (Barr and Asanovi, 2006), and operating system support (Marwedel, 2007). Unfortunately, for projects at this stage it is difficult and costly to change or optimize a system (Boehm, 1981). Thus, energy saving by resource substitution has to be addressed already in the earlier phases of software development (i.e., at the architecture or design level).

Software engineering research mostly focus on a specific resource but does not properly address substitution strategies at development time. In order to do so two major pre-requisites have to be fulfilled. We need mechanisms to analyze a system concerning its resource usage with respect to its target platform. This includes the prediction of resource usage as well as the indication of “weak-points” within the system. In a second step, we then need optimization mechanisms in form of patterns, anti-patterns or even idioms. In the long run the vision should then be “efficient resource usage by construction”.

Concerning estimation (Eskanazi,) presents a methodology for estimating memory consumption of systems built from source code components. He suggests the use of an auxiliary “provides” interface for each component to specify its memory consumption. (de Jonge, 2003) presents an approach to memory estimation based on the runtime behavior of components. The actual estimation is based on a careful examination of sequence diagrams and the combination of memory claims and releases of each component. (Murphy, 1998) presents an approach based on information extracted from class diagrams. Concerning energy consumption research mainly focused on low-level optimization.

Unfortunately, existing approaches focus on a specific property or resource and are not embedded within a development methodology. We therefore, have to extend existing methods such as the Unified Process (Kruchten, 2000) or KobrA (Atkinson et al., 2002). These approaches have to respect the iterative and incremental nature of modern processes (i.e., prediction has to become more exact as development proceeds), their use of modeling languages such as UML, and the recursive structure of modern, component-based systems. In summary, we need to know how a resource might be represented in diagrams at the design level. how resource interact, and how typical patterns concerning resource look like.

The next step is then to address optimization, or in other words how can we improve a system by introducing resource substitution strategies. One such approach is presented in (Balsamo, 2004). However, the approach does not use prediction or analysis techniques and does not take platform characteristics or process issues into account. One solution is to define patterns (solutions to recurring problems) on the architecture, design, and code-level (i.e., idioms) comparable to those presented in (Gamma, 2004). Of course, each pattern has to be evaluated in order to create evidence concerning its effects. Based on identified weak-points suitable patterns can then be selected and applied in order to establish resource substitution strategies within a system. In the long run one might even think about a compiler that provides options for optimizing code according to known context factors and strategies.

5.2 Dynamic Resource Substitution

Section 5.1 discussed how resource substitution can be realized at development or compile time. This might even include a pre-selection of a specific strategy. However, given the nature of mobile systems this might not be the “best” solution. Optimization of one resource (e.g., by using a specific substitution strategy) might have unwanted side-effects. Thus, improving energy consumption might increase CPU load or vice-versa. Therefore, means have to be developed that allow systems to adapt themselves at run-time. Run-time adaptation requires that a system is aware of its context and status in order to select the most appropriate strategy. To the extreme this might also mean “graceful degradation”. Here a system is degraded in such a manner that it continues to operate, but provides a reduced quality/level of service rather than failing completely.

In the context of model-based software development, we have to examine the environment and its varying context factors for defining the level of adaptability a system needs. Therefore, the acquisition of relevant information and flexible specification is a major challenge. In a second step, the chosen modeling language has to be adapted in order to support specification and translation of adaptive properties. Finally, standard mechanisms, including service brokers, facilities for (semantic) lookup, specification matching and service composition, context management and specific QoS functionalities need to be defined.

At the service-level (i.e., services are discretely defined sets of contiguous and autonomous business or technical functionalities) there must be an awareness of available resources and other service at runtime. Concerning self-adaptation there have to be means for lookup, negotiation and composition of services. Thus, the specification of a service has to describe both, functional and non-functional properties. As for the functional specification, a widely recognized specification technique is the component interface definition language (CIDL) (management

Group (OMG), 2001). A good survey on current work in specification techniques for non-functional properties can be found in (Jin and Nahrstedt, 2004). In the web-service community the web services description language (WSDL) is widely used. In order to automate the identification of the service capabilities, matching of specifications and composition of services, a semantic service specification is required. This, however, also implies the need for corresponding infrastructure mechanisms (i.e. semantic matchmaking).

Following the ideas presented in (Bastide et al., 2006) an approach is needed that supports the dynamic adaptation of system or component structures while preserving their behavior and services. This is especially useful when thinking about the dynamic redeployment of component services according to the available resources (e.g. CPU, memory). Another approach might be the design of a reflective architecture (Cazzola et al., 2004) in order to provides objects with the ability of dynamically changing their behavior using design information. Although both approaches are a step into the right direction they focus on adaptation mechanisms but do not provide support to actually install resource substitution at runtime.

In summary, dynamic resource substitution requires, beneath efficient strategies, systems that are aware of their context and are able to adapt themselves while preserving functionality and/or a specific level service quality (QoS). From the software engineering point of view we further have to think about adaptable architectures, adaptability specification, process integration, and quality assurance of such systems.

6 SUMMARY, CONCLUSIONS, AND OUTLOOK

Given the rising importance of computing systems and especially mobile devices, energy consumption becomes increasingly important. beneath extending the up-time of mobile devices, an increase in the use of natural resources should be avoided. Currents estimates by EUROSTATSpredict that in 2020 10-35 percent (depending on which devices are taken into account) of the global energy consumption is consumed by computers and that this value will likely rise even higher. Therefore, means have to be found to reduce this value.

The focus of this short-paper is on resource substitution as a means for energy saving. Therefore, the paper presented a selection of possible substitution strategies and discussed how to systematically use these within software development. Here we distinguished between development (static) and runtime (dynamic) approaches and presented initial ideas how this goal can be achieved. A realization of these ideas will lead to a significant energy saving which indirectly will protect our environment.

As discussed we have presented some initial ideas. Currently, we are preparing several projects in this regard that are aimed at performing extensive research in order to create evidence on the effectiveness of the proposed strategies, as well as methods and tools to support the development of energy-aware, mobile systems.

REFERENCES

Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J., and Zettel, J. (2002). *Component-based product line engineering with UML*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Balsamo, S. (2004). Model-based performance prediction in software development: A survey. *IEEE Transactions in Software Engineering*, 30(5).

Barr, K. and Asanovi, K. (2006). Energy-aware lossless data compression. *ACM Transactions on Computer Systems*, 24(3).

Bastide, G., Seriai, A., and Oussalah, M. C. (2006). Dynamic adaptation of software component structures. In *Proceedings of the 2006 IEEE International Conference on Information Reuse and IntegrationI*, pages 404–409.

Boehm, B. (1981). *Software Engineering Economics*. Prentice-Hall.

Buchmann, E. (2002). Konzeption und Implementierung eines Speichermanagers für ein konfigurierbares, leichtgewichtiges DBMS. Diplomarbeit, FIN, Otto-von-Guericke Universität Magdeburg, Magdeburg, Germany. in German, supervised by Hagen Höpfner.

Caracaş, A., Ion, I., Ion, M., and Höpfner, H. (2007). Towards Java-based Data Caching for Mobile Information System Clients. In *Proc. of the 2nd conference of GI-Fachgruppe MMS*, volume P-104 of *LNI*, pages 97–101, Bonn, Germany. GI.

Cazzola, W., Ghoneim, A., and Saake, G. (2004). Software Evolution through Dynamic Adaptation of Its OO Design. In Ehrich, H.-D., Meyer, J.-J., and Ryan, M. D., editors, *Objects, Agents and Features: Structuring Mechanisms for Contemporary Software*, Lecture Notes in Computer Science 2975, pages 69–84. Springer-Verlag, Heidelberg, Germany.

Chen, G., Li, F., Kandemir, M., and Irwin, M. (2006). Reducing NoC energy consumption through compiler-directed channel voltage scaling. In *Proc. of the 2006 ACM SIGPLAN conference on Programming language design and implementation*.

de Jonge, M. (2003). Scenario-based prediction of run-time resource consumption in component-based software systems. In *Proceedings of the th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction*.

Eskanazi, E. Estimation of static memory consumption for systems built from source code components. In *Proceedings of the 9th IEEE Conference on Engineering Computer-Based Systems*.

Falk, H. and Marwedel, P. (2005). *Source Code Optimization Techniques For Data Flow Dominated Embedded Software*. Kluwer.

Feeney, L. M. and Nilsson, M. (2001). Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. of the IEEE Conference on Computer Communications (IEEE InfoCom)*, Anchorage, USA.

Gamma, E. (2004). *Design patterns*. Addison-Wesley, Boston [u.a.].

Gurun, S. and Krintz, C. (2003). Addressing the energy crisis in mobile computing with developing power aware software. Technical Report 2003-15, University of California, Santa Barbara.

Höpfner, H. (2004). Serverseitige Auswertung von Indexen semantischer, clientseitiger Caches in mobilen Informationssystemen. In *INFORMATIK 2004, Band 1*, volume P-50 of *LNI*, pages 298–302, Bonn. GI. in German.

Höpfner, H. (2006). Update Relevance under the Multiset Semantics of RDBMS. In *Mobile Informationssysteme*, volume P-76 of *LNI*, pages 33–44, Bonn, Germany. GI.

Höpfner, H. and Schallehn, E. (2004). Anfragegeneralisierung zur komprimierten Repräsentation von Indexen semantischer Caches auf mobilen Endgeräten. In *Proc. of the Workshop on Foundations and Applications of mobile information technology*, number 4/04 in Preprints der FIN, pages 63–70. Uni Magdeburg. in German.

Ion, I., Caracaş, A., and Höpfner, H. (2007). MTrainSchedule: Combining Web Services and Data Caching on Mobile Devices. *Datenbank-Spektrum*. accepted for publication, to appear.

Jin, J. and Nahrstedt, K. (2004). Qos specification languages for distributed multimedia applications: A survey and taxonomy. *IEEE MultiMedia*, 11(3):74–78.

König-Ries, B., Türker, C., and Höpfner, H. (2007). Informationsnutzung und -verarbeitung mit mobilen Geräten – Verfügbarkeit und Konsistenz. *Datenbank-Spektrum*. in German. accepted for publication, to appear.

Kruchten, P. (2000). *The Rational Unified Process: An Introduction, Second Edition*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

management Group (OMG), O. (2001). Corba components - volume 1. Technical Report orbos/99-07-01, OMG.

Marwedel, P. (2007). *Embedded System Design*. Springer.

Murphy, G. (1998). Predicting memory use from a class diagram using dynamic information. In *Proceedings of the First International Workshop on Software and Performance*.

Pakdeepaiboonpol, P. and Kittitornkun, S. (2006). Low energy optimization for MPEG-4 video encoder on ARM-based mobile phones. In *Proc. of the 1st International Symposium on Wireless Pervasive Computing*.

Ren, Q. and Dunham, M. H. (2003). Semantic caching and query processing. *Transactions on Knowledge and Data Engineering*, 15(1):192–210.

Rudenko, A., Reiher, P., Popek, G., and Kuenning, G. (1998). Saving Portable Computer Battery Power through Remote Process Execution. *Mobile Computing and Communications Review*, 2(1):19–26.