Transaction Management for M-Commerce at a Mobile Terminal

Jari Veijalainen[1], Vagan Terziyan[2], Henry Tirri[3]

[1]*Department of Computer Science and Information Systems, Univ. of Jyvaskyla, e-mail: veijalainenj@acm.org*

[2]*Department of Mathematical Information Technology, Univ. of Jyvaskyla, e-mail: vagan@it.jyu.fi*

[3]*Helsinki Institute for Information Technology and Department of Computer Science, University of Helsinki, e-mail: Henry.Tirri@hiit.fi*

Although there has been a lot of discussion of "transactions" in mobile e-commerce (m-commerce), little attention has been paid for distributed transactional properties of the computations facilitating m-commerce. In this paper we first present a requirement analysis for m-commerce transactions, a graph-based transaction model, and a Transaction Manager (TM) architecture for a wireless application that protects m-commerce workflows against communication link, application, or terminal crash. The application interface, modules and log structure, as well as a pilot implementation of this TM for the location-based application are presented. We further discuss other alternatives to design such a TM that together can be called "Ontological Transaction Monitor", which assumes also monitoring constraints related to security and privacy.

Keywords: m-commerce transactions, terminal crashes, mobile commerce, workflows, personal trusted devices

1. Introduction

The main driving force for the rapid acceptance of small mobile devices is the capability to get services and run applications at any time and at any place, especially while on the move [1]. The experience from Japanese market shows that the most important factor is that the terminals are permanently carried around, and thus people can use so-called "niche-time" to use the gadgets for various things [2, 3]. The telecom industry estimates that there are now (winter 2005) 1.7 billion mobile users. According to some market analysis nearly half of the devices were internet-enabled in 2004 and the tendency is growing [4]. This means that at least 500 million, perhaps nearly one billion Internet-enabled mobile phones will be in use in the world in 2006. The number of these mobile Internet-enabled terminals, sometimes also called Personal Trusted Devices (PTDs), has exceeded the number of fixed-line Internet terminals around 2003 [5].

The term mobile commerce (m-commerce) is closely related with the term electronic commerce (e-commerce), both historically and conceptually. The definition of OECD for e-commerce is based on the concept of *electronic (commerce) transaction.* "An electronic transaction is the sale or purchase of goods or services, whether between businesses, households, individuals, governments, and other public or private organisations, conducted over computer-mediated networks. The goods and services are ordered over those networks, but the payment and the ultimate delivery of the good or service may be conducted on or off –line" [6]. E-commerce can be defined to consist of launching and performing electronic commerce transactions in the above sense.

M-commerce consist, correspondingly, of launching and performing *m-commerce transactions.* An M-commerce transaction is an electronic transaction that is conducted using a *mobile terminal* and a *wireless access network,* such as Wireless LAN, 2G or 3G telecom network, Bluetooth connection, or an Infrared connection. Notice, that this definition excludes those e-commerce transactions from m-commerce sphere, where the terminal is not mobile, albeit wirelessly connected, and the case where a portable terminal is connected with a fixed line to the network. On the other hand, laptops and even fixed terminals in cars, as well as all small portable telecom terminals and even Personal Area Networks (PAN) are included into the scope of the definition, if the conditions above are fulfilled.

In literature, also the term *mobile business* (or m-business) is used instead of m-commerce. We take the same view as the authors of [7, 8] that m-commerce is a subset of m-business, i.e. it consists of those activities where economic value is transferred from some party to another one as a part of the transaction.

Based on the above definitions, one can speak about m-commerce market and of its size, measured as the sum of the values of the m-commerce transactions. In Japan, the market size was 24 billion US-dollars in 2003, of which $7 billion

consisted of various mobile contents, including gambling, and 17 billions went into the wireless packet charges levied by the operators [9]. Global mobile commerce market - comprising concretely mobile entertainment downloads, ticket purchases and POS transactions - will grow to $88 billion by 2009, according to a Jupiter Research forecast [4]. About 50% of the market would be entertainment downloads and the rest ticket purchases. POS transactions would be worth of only $300 million in 2009. M-commerce transactions are an important class of applications on the PTD already now, and the above figures suggest that their importance will grow in the future. Thus, it is of high importance that the infrastructure offers proper security and transactional means to protect all actors in the environment against system crashes, but also against malicious actors and criminals.

In the above definition of m-commerce the notion of electronic transaction occurs. Looking at the context, the meaning of the term clearly points to a business transaction, as it is known in economics. Embedding the term into a particular technology is done from the perspective that information and communication technology mediates or facilitates these business transactions. The exact properties of the relevant technological artefacts are not at all addressed by the definition. Indeed, it is not very clear yet what kind of technical artefacts should correspond to the m-commerce transactions, as defined above. Taking a small step back into history, we can see a similar situation. The term "transaction" used to have almost ten closely related, but different meanings, already fifteen year ago, ranging from business transactions, over messages that mediate the contents needed in business transactions, to formal model of program execution within a database system [10, 11]. A transaction with ACID properties, i.e. an execution of a set of commands generated by an application or user within a DBMS in a serializable and at least recoverable manner [12] is conceptually and practically certainly something else than buying a book using a mobile terminal. Still, they have the commonality, that with a rather high probability the e-commerce system recording the customer order and other steps in the book delivery process uses one or several DBMS and their transaction processing capabilities to guarantee unique view on the responsibilities towards the customer. Thus, m-commerce transactions should somehow subsume traditional DBMS transactions, but evidently, this is not yet enough, because there is the wireless tiny terminal, wireless access network and often a portion of Internet that are playing a role as facilitators of the m-commerce transactions.

During the late 1970´s and early 1980´s, it was already recognized that the transactional properties (or something very similar) are needed for distributed computations in general. The centralized transaction model was first extended to support distributed transactions in a distributed database context [12, 13, 14]. Soon it became evident that distributed transactions could not be reasonably used in autonomous environments or in such environments where the transactions last hours or even weeks. Many "advanced" transaction models were developed for these environments and many of them are published in [15]. More complicated models present individual transactional computations as trees with height larger than one. One modelling dimension is the selection of the correctness criteria that divide the computations, modelled as linear operation histories or trees, into acceptable and non-acceptable ones. A standard way of doing this is to set up an equivalence relation among histories or trees and classify those, which are equivalent to a serial history or serial forest as serializable, i.e. acceptable. Check e.g. [16, 11] for a more complete analysis of diverse transaction modelling incentives.

Is m-commerce an area that would again need its own transaction model? Isn't the work of MeT already performed enough? Something called mobile e-commerce transactions have been indeed developed among others in an industry-led consortium called Me-forum, later MeT Ltd [17]. This effort has produced public white papers [5] where the opportunities and risks of m-commerce are discussed. Scenarios (business models) for five subtypes of m-commerce have been developed. Protocols to support them are defined in technical documents of MeT. On June 12th, 2002, a larger consortium called Open Mobile Software Alliance (OMA) was announced [18]. Its goal is to create a truly global and interoperable m-commerce market. The key technical goal is end-to-end interoperability at the service level and thus end-to-end transactional properties of the services should also be considered. MeT Ltd will cooperate closely with OMA. They have a cooperation agreement [18].

In a closer look the need to go beyond individual messages and message exchanges, i.e. protocols, becomes more than evident, because the overall business transaction can consist of several interactions with different actors, such as merchants, financial institutions and logistic companies; interruptions between the phases can cause for example the order to be accepted but the goods not been paid, or goods been paid but not delivered, etc. The infrastructure should offer mechanisms that help in avoiding these. Such research was considered vital e.g. in Asilomar report [19].

Another issue is the security and trust closely related with the infrastructure and mobile terminals. Unless people feel that m-commerce infrastructure is secure and protects their privacy they do not want to use it. Therefore, security and privacy should be combined in a new way with the transactional mechanisms into an integrated environment [20]. Appropriate risks are discussed e.g. in [21, 22]. These aspects are addressed also in [5], where the transactions are divided into remote, local, and personal. The remote transactions are performed over a digital mobile public network, local ones over a local link, and the personal ones among devices controlled by the user.

We exclude in this context the personal transactions in the above sense, because a person cannot reasonably have commercial transactions with him- or herself. Let us from now on view the M-commerce transactions from the technical point view, consisting of computations and communications facilitated by the infrastructure. In this view they are inherently distributed, because there are parts running at different computers and these are tied together over a wireless link by a communication protocol. From the structural modelling point of view, they can be viewed as a special kind of *workflows*. In this, more formal structural view, an m-commerce transaction refers to:

- a specification of a m-commerce workflow composed of interoperable specifications at different actors;
- enactment of the specification by the distributed m-commerce infrastructure, comprising the execution of the relevant protocols and the local steps launched by the protocol message exchanges at different players.

The properties of m-commerce transactions are evidently different from the traditional centralized and distributed database transactions [12, 14]. The same is true also for many "advanced" transaction models developed (e.g. [15, 16, 11]), although some known transaction models are designed for application environments with similar properties as m-commerce environment. Here of particular relevance is the S-transaction model [10, 23] developed for international banking environment with strong autonomy properties. Sagas [24] and nested sagas [25, 26] are also of relevance, but as a special case of S-transactions they do not need to be paid a special attention to. The most relevant work is reported in [27] where the workflow specification, the transaction specification and execution graph are closely tied together. We analyze below more closely the commonalties and differences of m-commerce workflows and those in [27].

One of the most important developments from m-commerce transactions point of view is that the terminals are being developed towards Personal Trusted Devices (PTDs) containing private keys, private and corporate information, and perhaps also credit card information and wireless cash. Stealing or misusing such a device or information carried in it can cause great damage for the device owner and other parties involved.One of the starting points of our work is to design such a transaction model and corresponding transactional mechanism in the m-commerce environment that is intertwined with security, privacy, authentication, and authorization mechanisms (cf. [5]). As special e-commerce transactions, m-commerce transactions complying with the model should guarantee the atomicity notions introduced in [28] for e-commerce, which as such can be formulated as special kind of semantic constraints between subtransactions of S-transactions (cf. [10, 23]).

In the sequel we deepen the above analysis about the need and form of transaction modelling for m-commerce environments. In section 2 we relate business models and transaction modelling concepts with each other. This is done by analyzing two of the five business scenario types suggested by MeT [5]. Based on this analysis we refine transactional requirements and properties, as well as compare them with known transaction models. In section 3 we describe a more complete transaction model for m-commerce satisfying the requirements and deduce and analyze more in detail the properties of this transaction model. In section 4 we discuss shortly the implementation aspects of a simple transaction manager. In section 5 we look for a more sophisticated design. In Section 6 we give some samples of related work. Section 7 concludes.

2. Business models for m-commerce

In our earlier work [29] we have shown that the global m-commerce environment should be viewed at least from four perspectives. One of them is Business Models. They determine how the business transactions are specified and when and how the players interact with each other. The business transactions are an abstract representation of the m-commerce workflow specifications.

One way of defining a business model is given in [30]:

- an architecture for the product, service and information flows, including a description of the various business actors and their roles;
- a description of the potential benefits for the various business actors;
- a description of the sources or revenues.

We analyse below some concrete business transaction instances that are simultaneously m-commerce transactions in our sense.

2.1. Mobile e-commerce actors

We first look at Japan, because the wireless Internet (e.g. i-mode™[1]) has existed there since February 1999 and certain m-commerce actor categories have emerged and are economically viable. According to [2,31] in the market initiated by NTT DoCoMo one can distinguish the following actors: a user; mobile network operator (MNO); telecom operator; application

[1] *i-mode, mova, and FOMA are trademarks or registered trademarks of NTT DoCoMo, Inc. in Japan and other countries.*
ToruCa is a trademark of NTT DoCoMo, Inc. in Japan.

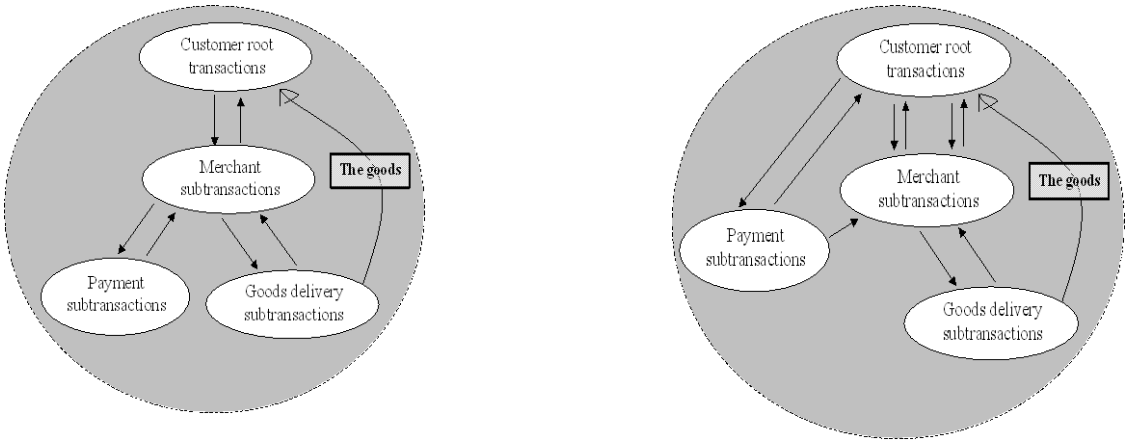
provider; facility supplier; information provider; contents holder; solution provider; financial institution; terminal manufacturer. Not all these actors take part in actual m-commerce transactions, but get their revenues indirectly. Typically, companies providing the network infrastructure or system software for the terminals are such players. Typical of Japanese market was around 2000 that services are billed and thus payments are not performed on-line [31]. Thus, conventional telecom business models were first adopted, rather than those applied in Internet environment. Later studies show that the proportion of especially credit card payments in mobile commerce has increased in Japan after 2000 [32, p.123].

2.2. Internet e-commerce over PTDs

The basic idea in this paper is that the wireless network is one of the access networks to the e-commerce infrastructure offered over the Internet. The PTDs are only a new terminal class through which the e-commerce transactions can be conducted. The capabilities of the PTDs are (much) more limited than in ordinary PCs or laptops (see [33]), although the mobile terminals are rapidly developing and the top models are programmable, have a rather large memory (200 MB), and have a rather fast processor. The interaction patterns between the customer and other players can basically remain the same as in the cases, where the business is conducted over PCs. Thus, one can consider e.g. the eleven business models analysed in [30] to be used by customers over PTDs. Is this realistic in all cases and when does it make sense? To answer one must remember that the interface facilities of the PTDs are not capable of showing fancy graphical layouts and complicated forms. They must be redesigned and described using e.g. WML [34] or c-HTML – or XML-related mark-up languages, especially XHTML [5, 34]. In addition, some scripting language like WML Script is needed. Second, in contrast to Internet, users must pay for the data transfer in MNOs' networks and transferring data is relatively slow – and expensive, although the price is going slowly down.

To give concrete examples we look at three business transactions in [30]. One is E-shop with credit card company as financial institution, the other one is Info-brokerage type of service with direct on-line payment, third is wireless banking (see [35, 36, 37]).

In Fig.1 a) the E-shop case is schematically depicted. From m-commerce transaction point of view we omit the goods selection phase that might involve several message exchanges. The m-commerce transaction begins upon placing the order. Thus, the *Begin-tr* clause is placed at the terminal in front of issuing the order message. During the order processing, the merchant contacts the credit card company and checks whether it is ready to allow the purchase. After checking the inventory, the merchant either acknowledges or denies the order. If tangible goods are ordered, the delivery logistics is asked to deliver the goods to the address given by the customer.



a) Implicit payment b) Explicit payment

Fig. 1. E-Shop and E-Service with different type of payments

In the case of intangible goods, like music, the merchant can deliver the goods over network by push or pull approach. In the latter case, usually the customer gets an email that contains an URL, from which the goods can be loaded [38]. This additional information flow facilitated by email is not shown in the picture. It is also worth of noticing that the PTD does

not need to be the delivery address of the intangible goods (see [39]), but the target address can also be another (home) device capable of storing/presenting the contents (e.g. video recorder).

The second case is depicted in Fig.1 b) The difference to the previous case is that after the customer has placed the order, the merchant replies with a special form that contains the payment information for the bank or other payment service. The customer then contacts the bank and pays the goods or service using funds transfer from her account to the account of the merchant. For authorisation, the Personal Identification Numbers (PIN) of the customer issued by the bank are used. After a successful payment the customer gets another form from the bank that she sends in form of a request to the merchant. It contains the payment details.

Usually, there is an additional information flow between the payment service and merchant to inform about the payment (shown in the figure through an arrow). The case is described more elaborately in [39].

In the banking case there are only two explicit players, the customer and the bank, such as Nordea [35, 36]. One can access Nordea bank's services through two functionally equivalent but externally different WWW pages. On the front page one can select the interface. Thus, the user can do all banking operations using an IP-enabled wireless PTD or usual PC. The actual server uses secure end-to-end connections (https). The operations include all typical operations for domestic and international funds transfers from customer's own account. Nordea bank offers an even simpler banking interface for pure WAP devices [40]. The interaction patterns remain the same as above, but the forms exchanged between the client and the server are adapted to the tiny resources of the current generation of WAP phones. The authentication and authorisation is in all cases above based on lists of use-once PINs.

Looking at the technical requirements, with the currently applied wireless tariffs that are either connection time or data volume based or both, an obvious requirement for the m-commerce transactions is: *the transactional protocols should run as quickly as possible and move as little data as possible from and to the PTD.* In practice this means that the m-commerce protocols should exchange as few as possible messages and as little as possible data over the wireless link. This raises immediately the question, whether the interaction patterns designed for the "cost free" Internet infrastructure can indeed be used as such, or should they be redesigned for m-commerce applications

On the other hand, as was inferred in [20], the interaction pattern of the m-commerce transactions should be the more complicated the higher value the business transaction has. This is because increasing the number of end-to-end authorised message exchanges during the transaction execution makes it less probable that a fraudulent person would be able to run successfully the m-commerce transaction using the PTD. This requirement is evidently in contradiction with the above minimisation requirement. For example, for a digital map worth of 5 euro the transmission cost is more relevant than for a 200-euro book.

2.3. Location-based services; the taxi example

These services are more in the domain of a MNO than the previous ones [41]. The basic idea is that the terminal has a position on the earth and this is made known to applications running on the infrastructure. The infrastructure can be running on MNOs' sphere of control or at some external service provider. A typical query is: "Where am I now?" "Where is the cheapest restaurant that is 500 m away from my position?" A very useful service request is: "Send me a taxi right now!" For further discussion on the emerging applications see e.g. [42].

Displaying the WGS-84 coordinates, or using maps, voice, etc can answer the first and very basic query above that is implicit in any location-based activity. The coordinates can be generated by a GPS-enabled terminal itself or provided in whole or in part by the network infrastructure. If the terminal is not GPS-enabled or is outside GPS coverage, it must communicate with the base stations in order to collect the data that is necessary for calculating the position. The calculation can be done either by the network or by the terminal.

The general logic of the terminal-initiated location-based service requests is the following:

[Terminal-> positioning infra: locate_me (MyId,Qid)
positioning infra-> terminal: (MyId,Qid,XYZ)]
Terminal: form the queryQMyId,(QId, XYZ,otherPara)
[Terminal ->LBS appl.:query
Q(,MyId,QId,XYZ,otherPara)
LBS application -> terminal
Result(,Qid,ResultPara))]

By "[", "]" we denote here the borders of subtransactions that must be performed in their entirety, i.e. in an atomic fashion(see below).

Let us now look at the taxi-ordering example. It differs from the above digital content delivering examples because a physical taxi comes to the location where the customer is (or was when the taxi was ordered). We can use both of the ways above to perform the location query *Q*. Let us take the former form. See the following taxi ordering scenario from the terminal perspective:

[*Terminal-> positioning infrastructure: locate_me (MyId,Qid)* % 1. subtransaction start
positioning infrastructure-> terminal: Result(Qid,XYZ)
]% 1. subtransaction end

[*Terminal: form the query Q(MyId, XYZ,Qid,service provider=?)* % 2. subtransaction start
Terminal -> Global/local Service Directory: Q(MyId, Qid, XYZ, Service-type="taxi", Service-provider=?)
Global/local Service Directory-> Terminal : Result(Qid, Service-type="taxi", Service-provider=URI_local_taxi_list)
] % 2. subtransaction end

*[Terminal: form the query Q(MyId, Qid, XYZ ,otherPara) %*3. subtransaction start
Terminal ->Service-provider(URI_local_taxi): Q(MyId, ,Qid, XYZ,otherPara)
Service-provider(URI_local_taxi) -> terminal Result(Qid,Resultpara)
Terminal -> Service-provider(URI_local_taxi): ackQ(Qid)
Taxi arrives to the place (XYZ) after W minutes from placing the order.
] %3. subtransaction end

[Customer is transported to the target address % start of 4. subtransaction

Customer pays the trip wirelessly using a digital credit card within the PTD (Mobile Wallet)
] % end of the 4.sub transaction

One of the problems that have not been generally solved in the location-based service area is how to discover the suitable local service provider that the customer needs for a certain service like the taxi service. The directory service address(es) should be resolved before the actual services are used. This can be done in several ways ranging from manual search and configuration to various directories and search engines that the Internet infrastructure offers to the terminal base. One logically simple, but very challenging from implementation point of view, solution is to offer a global directory with fixed, globally known address that maps the customer coordinates and service type to a local URI that provides that type of services. Other solutions would use a more local directory and perhaps also the user identity (MyId) that can be the current IP-number and/or the phone number of the customer. In addition to the service address, the exact interface of the service should be obtained as part of the service provider description.

When the service provider address is resolved, a service request can be compiled adhering to the interface specification of the local service and sent to it. The reply from the local taxi provider can contain in its *Resultpara* part several results. Result(Qid,Resultpara) can say "Taxi nr. 999 comes to <address> in ca. 4 minutes; order Nr 666"; or "Sorry, we cannot deliver a taxi within 4 minutes, but within ca. 15 minutes". The ackQ_(Qid) is in this case important, because through it the customer commits to the order. Sending Nack_Q(Qid) cancels the order. The parameter *otherPara* in the request above must in this case contain e.g. the time limit, during which the taxi is expected to arrive. After the customer has committed to the order, he is primarily expected to wait for the taxi at <address> for the agreed upon number of minutes W. If W exceeds the time in the Result, the customer should not need to pay, even if she does not wait. If the taxi does come on time to <address>, but does not find the customer there, then the customer should pay (see [43] for more on disputes in e-commerce). The commitment to an order and disputes are tricky aspects that are dependent on business habits in a country and contribute to the roaming heterogeneity. The differences have influence on the transaction borders and are in this respect also relevant in this context. These are for further study.

In Fig.2 all the actors and subtransactions involved in the taxi example are shown in a UML diagram style. The actors are described on top of the figure and the arrows are message exchanges between the actors. There are four different transactions, as perceived by the customer/terminal, corresponding to those in taxi ordering scenario above. As it can be seen, the customer transaction spawns new subtransactions at other actors that are neither visible nor controllable for the customer in cases 3 and 4. They are, however, necessary for the success of the transaction started by the customer. Thus, subtransaction 3.2 (positioning taxi cars in the vicinity of the customers) and 3.3 (suggesting to a particular taxi car that it would serve the customer and the acknowledgement) are necessary subtransactions that are needed before the confirmation 3.4 can be sent to the customer in transaction 3. In the payment transaction 4 the initiator is the car by request for payment

4.1. The terminal returns the credit card information over a wireless link (Bluetooth, infra-red, RFID reader; cf. FeliCa®[2] [52]) and the taxi car sends this further to the credit card infrastructure in order to get confirmation for the transaction. Once this arrives, the details are stored persistently and a receipt is issued to the customer (4.3). The latter can be digital or physical; i.e. a paper receipt.

From the customer/terminal point of view the precondition of the payment is that the taxi indeed provides the service, i.e. transports the customer to the target address. The payment action starts with the delivery of the credit card information, together with the sum to be paid and the used currency, and ends with the reception of the receipt.

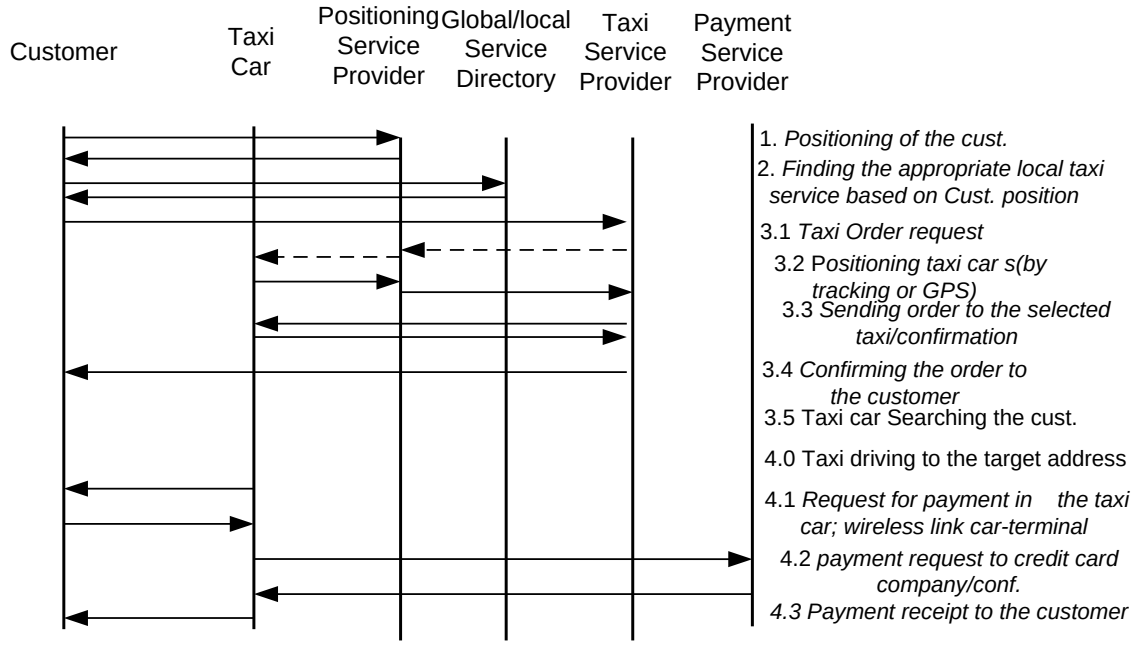


Fig. 2. Taxi-ordering as an example of m-commerce transactions (adapted from [45], Fig. 8])

The above taxi service requires privacy protection measures and customer trust, too. For instance, the tracking of phone number should only be enabled for this particular purpose, taxi-order here and now, if there is a need to set up a voice connection between the customer and the taxi driver In general, m-commerce transaction security can probably be improved through location-based authentication (see [46]). Location, as calculated from a location signature, adds a new dimension to user authentication and access control. It can be used to determine whether a person is attempting to log in from an approved location or performing tracking from an allowed region, and using approved services.

2.4. Business models with MNO as proxy

Because wireless communications are relatively slow and expensive, and the capabilities of the PTDs are limited, one is easily led to the following idea. Another component, *proxy*, could present the terminal and thus the customer against other actors. The terminal could submit a request to the proxy that then acts on behalf of the customer controlling the PTD, based on the instructions given in the request. Basically, in any m-commerce business model one might try to add this kind of a middleman. Looking this from the m-commerce workflow point of view, a proxy means making the execution tree (or rooted TAG) higher but narrower at the first level. Typically, such a proxy can perform searching, information gathering, sorting of the received data etc. This saves a lot of bandwidth over the wireless link, costs, and processor and memory capacity of the PTDs, as well as energy at the terminal.

A more general business model using a proxy is the one, where the proxy is explicitly equipped with *capability* from the customer. The capability should contain the instructions for proxy and it should be signed by the private key of the customer. It should also contain the certificate issued by a Certification Authority. The clearly missing piece is a general language to describe the assignments from the customer to the proxy. Technically, the language should preferably be based

[2] *FeliCa is a registered trademark of Sony Corporation.*

on XML as BizTalk™ [47] is. Another possibility is emerging from the Semantic Web community through goal concept in WSMO [48] that could be viewed also as an assignment description. In the ASG project an "attraction-booking" prototype has been developed for LBS area that is based on this kind of middleman architecture and goal concept [49]. This kind of a *mobile assignment/contracting mechanism* is left for further study.

3. Towards m-commerce transaction model

From the above analysis and examples one can draw the following conclusions. The request-response pairs observable at the customer terminal can be understood to delimit *steps* or *actions* of a workflow. The workflow instance at the terminal consists of a sequence of these + local computations. Further, in all m-commerce workflows the user is the ultimate source and initiator of the workflow and has the ultimate control over the workflow. The workflow view can be expanded to the other actors, too. Their activities can also be understood as subworkflows of that running at the customer side. In the "traditional" e-commerce transactions presented in Fig.1 a) the merchant performs two steps (payment and goods delivery) that are triggered by the customer order. In the taxi example the taxi service provider performs first positioning of taxi cars in the vicinity of the customer with coordinates (X,Y,Z), selects a taxi, and sends on order to it with detailed positioning information. Thus, a natural way would be to perceive the workflow structure as a hierarchy. We can further see that a subworkflow is always started by a request from one actor to another one. Thus, the structure can be viewed to have a spanning tree. This can be defined by modelling each (sub)workflow at an actor as a node of the tree and drawing an arc from a node to the other a directed arc for each request. By adding for each response and acknowledgement, as well as goods an arc in the direction it travels between the actors we get a *workflow execution graph*. The graph models the execution of an individual workflow instance.

Figures 1 and 2 are two different representations of this graph class, for different workflows.
It is evident from the examples above that there is not just one m-commerce workflow type that the terminal should be able to run, but many. They differ in topology from each other, but they all have evidently a spanning tree in the above sense. So, the workflow execution graphs can be viewed as Rooted Directed Acyclic Graphs (RDAGs). In general, the height of the spanning tree can be any finite number.

In spite of the common abstract structure, there are differences in instances, even if the execution graph would be equivalent for two instances. This is because different workflows might require different kinds of protocols to be run against the other actors. Taking into consideration the local business habits and specifications even the "same" abstract workflow, such as "ordering a taxi" – might require different concrete protocols to be run at the terminal in different countries - because e.g. the local taxi service interfaces might require different protocol data units at application level.

After establishing the hierarchical common structure for m-commerce workflows, one can ask, what would the attached transactional concepts and properties be? If we look at the examples above, we see that in both cases, the entire workflow should run from the beginning to the end in order to satisfy the customer needs. The beginning and the end must be defined in the *workflow specification.* In the Fig. 1 b) the workflow starts with the order and ends when the book arrives and the customer keeps it; or the customer returns the book to the merchant maximum X days after the reception (notice that X is different in different countries in Europe). In Fig. 2 the workflow begins when the customer positions itself and after it starts seeking the taxi service. The workflow ends when the customer has reached her destination and has paid. The workflow can, however, break at several points. First, positioning does not work (subtransaction 1 fails); second, automatic taxi service cannot be found (subtransaction 2 fails); third, for some reason no taxi with suitable properties can be found (subtransaction 3 fails); fourth, the customer is not found by the taxi driver (physical action fails); fifth the taxi is not able to drive to the target address at all or is too much delayed (physical action fails). *So, we see that the workflows should have atomicity notion that ties together all actions in the instance and covers both physical actions and computerized actions.* The entire workflow instance can thus be usually treated as *an atomic transactional entity* (either all necessary actions should be performed or none).

As the examples above showed, it makes sense to introduce subtransactions into the overall transaction. This is because it can be that several actions are needed in order to achieve a larger sub-goal, and because the subtransactions can be recovered independently from other subtransactions. They are therefore atomicity sub-units. Their exact properties must be analysed during the workflow specification time. Some of them can be idempotent, some can recover forward, some backward. For instance, the positioning subtransaction is idempotent, because it does not matter how many times the positioning is performed for the end result. Searching for the suitable taxi service can also be performed arbitrarily many times. Taxi or book order should be done exactly once; otherwise more than one book or taxi can arrive. Also the payment should happen exactly once in both cases.

Finally, the smallest atomic units are the actions introduced above, i.e. request-response pairs. They should either happen completely or not at all. As we have seen above, an action at higher level can spawn a transaction at the next level in the execution hierarchy. The actions 3.2-3.3 above must both succeed, in order for the action at the root level to succeed.

The m-commerce transaction model is thus a reasonable concept. Atomicity is a central goal of this model, as is the case also for S-transaction model [10, 23]. The PTD representing the customer is the ultimate source of the activity and thus naturally the *root* of the transaction. It must often request several tasks or steps to be performed at different actors (ordering, positioning, paying) over a wireless network. Thus, there are *subtransactions* involved. *M-commerce transaction instances are distributed and they can be modelled as Rooted Directed Acyclic Graphs (RDAG).* The subtransactions are in some cases directly traditional DB-transactions with ACID properties, e.g. funds transfers (cf. [27]). There are also contractor-subcontractor relationships present in the environment (logistics, searching, indirect payments) that necessitate a deeper *subtransaction hierarchy.* Therefore, akin to S-transactions the root at the beginning of the execution cannot determine the depth of the hierarchy. An important aspect is that m-commerce transactions often rely on *existing infrastructure that remains as it is.* Therefore, the root transaction does not necessarily know how deep the hierarchy/DAG actually is. The hierarchy can vary from business model to business model and even from transaction instance to transaction instance. Therefore, the *controllable scope of m-commerce transactions is the root and first level of the transaction execution graph* no matter how large the entire RDAG of the m-commerce transaction is. This first level forms also the scope of standardisation by MeT, because the deeper levels already exist and their design cannot be much influenced. The m-commerce environment is highly autonomous and heterogeneous and has legacy systems running. This has several implications. First, the m-commerce workflows are typically inter-organisational. Second, they are in a new way dynamic, because the terminal can roam to any place in the world and initiate such an m-commerce transaction. Consider in this respect a roaming customer and the taxi example above. Evidently, the PTD should be able to communicate and interact with the *surrounding* infrastructure in order to be able to order the taxi. What guarantees, that the PTD is able to do it exactly in the same manner in New York, Istanbul and Beijing? This is a new legal, business model, organisational and hardware/software problem. We call it *roaming heterogeneity.* This phenomenon was a reason for establishing MeT [5] and later OMA [18]. The roaming heterogeneity and existing global infrastructure is a difference to S-transactions as well as to the environment in [27]. In order to mitigate the above problem and to cut the data transmission costs one must try to minimise the number of parties, which the root must interact with during an m-commerce transaction. Bandwidth and computational restrictions at the terminals are also important. *Thus, m-commerce transactions should be as "narrow" as possible at the top*, i.e. they should access as few as possible other actors.

Like S-transactions, M-commerce transactions often contain *real actions* [13, 10] as part of them (delivering tangible or intangible goods or non-digital services like taxis). Sometimes the real actions can be technically tied to the running m-commerce transaction; sometimes the user must tell the terminal of the occurrence of a real world action (like arriving of the ordered book, receiving the physical receipt).

Similarly to S-transactions, *atomicity preservation* is the key property of the m-commerce transactions. Both try to enforce an *atomicity constraint* [10] that says that either all necessary "positive" subtransactions are performed or otherwise they are cancelled. The constraint sought to be enforced usually fulfils certified delivery [42], but this is not always enough. Due to the unreliability of the E-commerce infrastructure and real actions involved, atomicity constraints can only be enforced with certain probability that is less than one. The difference to the S-transaction model is that not even ACID properties can be required at leaf level of the execution graph. Therefore, the atomicity can only be achieved with certain probability. Further, the transactional protocols cannot be used and the corresponding properties not enforced throughout the execution. In contrast, the S-transaction model requires that within the execution graph each parent-subtransaction protocol is the same well-defined protocol with request-response-ack/nack message exchanges.

Durability, Consistency and Serializability are important concerning the subtransactions of different m-commerce transactions at one actor, but they are not the key properties of the overall m-commerce transaction. In fact, they play very similar role as in S-transactions [10, 23]. *Permanence* of the state is also of high importance for the root transaction running at the terminal. Because the transactions can last days or even months (trading tangible goods, ticketing) special measures must be taken to store the transaction states (logs) persistently.

As e-commerce transactions, m-commerce transactions can involve *cancellations and dispute resolution.* How they are handled depends largely on the business model and local business habits. As such, the problems and solutions are similar to the Internet e-commerce cases, if Internet e-commerce is performed over a wireless link. A new problem is location-related disputes and cancellations (taxi did not arrive to the right place at right time) and should be studied further.

A very important new thing is that authorisation; authentication, security, and privacy aspects are pervasive in the m-commerce transactions [39, 20]. This requires the user-PTD interactions to be modelled and logged as part of the root transaction. Also, encryption and transactions must be more closely brought together.

4. Implementation considerations

As said above the controllable scope the m-commerce transactions are the root and the first level subtransactions. Because the latter are running at e-commerce or infrastructure servers, we assume for simplicity that they are implemented using DBMS with full transactional facilities. Thus, such local subtransactions have ACID properties at the server. The same holds for the subtransaction trees rooted at the servers.

4.1. Transactional functionality at PTD

The main challenge is how to support m-commerce transactions at the scarce-resourced PTDs. Can workflow specifications be made executable at PTDs? In theory yes, but in practice this would require much resource from the terminal. We do not believe that the current or next generation of PTDs would be able to run an interpreter for general workflow specifications. On the other hand, the top-end terminals (such as Nokia 9500 and 3G terminals in Japan) have hundreds of megabytes memory and are since many years capable of running Java so it is only a matter of time when this will also be possible.

We assume in the sequel that there is an application that materializes the m-commerce workflow root functionality and that runs at the terminal, i.e. the logic presented in the taxi ordering scenario example above is indeed programmed using Java. There can be several such applications hosting one or several workflows or only one (the workflow interpreter). As we see from the examples, the role of the PTD in transactional sense is somewhat similar to a 2PC co-ordinator. Both must keep track of the state of the subtransactions and guarantee that their end state remains coherent, although the criteria for the coherence are different for 2PC and MC Transaction Manager (MCTM).

What are the threats that the MCTM should guard against? First, the *action atomicity* can be jeopardized by communication crashes and/or terminal or server crashes. That is, typically a request is sent, but a response never arrives at the terminal. This is not a pure communication problem, because the request might have caused a state change at the server (e.g. order is placed) and thus simply e.g. re-issuing or forgetting the request will not be appropriate. This is really an application (and protocol) design problem, because it must be able to decide when it makes sense to re-issue or cancel the request after a particular crash. Both the applications running at the terminal and at the server must be “recoverable” in the sense that they recognize that a crash happened and remember what was done before the crash.

Subtransaction atomicity is jeopardized, if the sequence of actions is not complete in a semantic sense. If the two interactions with the bank/credit card company above in Fig. 1 or Fig. 2 are interrupted at any point of time where both of them have not been completely performed, the subtransaction atomicity is not preserved.

Finally, the *transaction atomicity* is jeopardized if the subtransaction atomicity is jeopardized or some semantically necessary subtransaction is missing. If the merchant is not informed of the successful payment, the customer will most probably not get the goods. Notice that lacking atomicity at a lower level constituent implies lack of next higher-level atomicity, but not vice verso.

From these follows that MCTM running at the terminal must be able to distinguish and *log*: 1) beginning of m-commerce transaction BegTr, 2) end of it EndTr, 3) beginning of a subtransaction BegSTr, 4) end of subtransaction EndSTr. Assuming that a subtransaction corresponds to a sequence of actions, i.e. request-response pairs towards the same server while performing a service, MCTM further needs to mark 5) start of action BegAC, and 6) end of action EndAC.

Distinguishing between different m-commerce transactions and their subtransactions requires a *unique naming scheme.* This is in a natural way hierarchical (TID.STID). The name should also contain the technical identity of the actor (URI), at which the subtransaction is being run, in order to recover the action/subtransaction, if necessary.

Further requirement is that the subtransaction borders must be determinable during the execution because the m-commerce transaction can evolve in different ways based on the user's decisions or application logic.

How can the PTD decide upon the above borders? A basic facility the PTD must offer is *transactional context* that either the user or the (micro) browser can ask the PTD to *enter* or *leave.* This is required for two reasons. First, the PTD is used for many purposes, and not all are related with m-commerce transactions. Second, not all subtransactions or actions of an m-commerce transaction protocol need be included into the scope of an m-commerce transaction (e.g. browsing catalogues). Entering the transactional context means that the application/browser begins to interact with MCTM to insure transactional properties.

Including certain subtransactions or actions dynamically into the transactional scope or excluding some of them dynamically and possibly conditionally can be solved by letting the user to set the m-commerce (sub)transaction borders. If we accept this paradigm in this environment, then the mobile user must indeed set the transactional borders. Concretely, this would mean that while in the transaction context she is offered e.g. a menu, where she could issue the above actions. This approach has far-reaching ramifications, as the entity responsible for the correctness of the m-transactions would be the user.

Another way is an implicit transaction border setting by the application software at the terminal. The latter requires that the software can recognise when the user has initiated such an action that should start an m-commerce transaction or subtransaction, and which action or incoming message should close the m-commerce (sub)transaction. This is possible, if the application knows which actions belong to the subtransaction. Subtransaction can also be closed when the application or user wants to start interaction with a new server. The same holds if the incoming message contains TID or STID or information based on which the correct transactional context can be deduced. This requires protocol with memory from the server. The latter requirement is emphasised because the m-commerce transactions are typically long lasting. Thus the user must be allowed to run several m-commerce transactions simultaneously. As a consequence, an incoming message does not necessarily belong to the currently running m-commerce transaction.

The simultaneous execution of several m-commerce transactions, and also terminal crashes, require that the PTD offers operations *suspend(<TID>)* and *resume(<TID>)* for the user. Using them, the user can switch between different transactions within the transaction context or leave it and return to it, when appropriate.

There are two possibilities for the transaction border control: either the user controls the m-commerce transaction borders or the PTD application software does it. We can also combine the solutions into one, i.e. if the application does not know where the borders should be placed, it asks for user's assistance, otherwise it does it. The applications running at the terminal must be quite sophisticated in order to fulfil the above requirements. They should be prepared for diverse crash situations and be able to decide when to recover forward or backward after a certain kind of a crash. Should this not be the case, the transaction histories or traces should still be retrievable from a persistent memory so that the user manually checks what has happened and can then decide whether some further measures are needed, such as contacting an e-commerce site. Whether application or the customer or both take the corrective actions, at least a persistent log must be produced by the system, from which the history prior to the crash can be retrieved.

4.2. Design principles of a simple MCTM

An M-commerce application (workflow root) execution at the terminal corresponds to one instance of M-commerce transaction, as perceived by the MCTM. Consequently a unique Transaction Identifier (TID) identifies it. The logical design of TM is as follows. Applications will interact with MCTM through a standard, programmatic interface allowing them to start and end a transaction, subtransaction and action. Further, applications can write a CHECKPOINT and retrieve any of the earlier stored items at any time. The TM basically stores the items handed over by the application into a persistent log in the order they arrive. The persistent log is implemented using the most persistent (RAM) memory the terminal offers. In the top-end terminals we can usually rely on file system of the platform while managing the persistent data.

For the latter purpose the application must store as part of the END_AC record also the name of the compensating action and parameters with which this must be executed in order to reverse the impact of the action being closed. The action might also be such that it does not need to be compensated or it might be non-compensatable. We envision that often the application must ask user what to do in case of a backward recovery.

A more detailed TM architecture is presented in Fig.3. The Dispatcher that keeps track of the TIDs etc. and is able also to analyze the state of the transaction implements application Interface. Archivist handles the log i.e. Application Log Memory (ALM). Archivist processes BeginTR, EndTR, BeginSTR, EndSTR, Begin AC, and EndAC by assigning Ids for appropriate transactions, subtransactions and actions, storing corresponding log records with appropriate timestamp values in the ALM. It also stores the CHECKPOINT information. We assume in this version of the MCTM that the application writes the actual checkpoint data into a persistent store (file) and only the reference (file name) is stored into the ALM, along timestamp. This is facilitated by the SetCKPNT operation. It is up to the application to interpret the file contents properly.

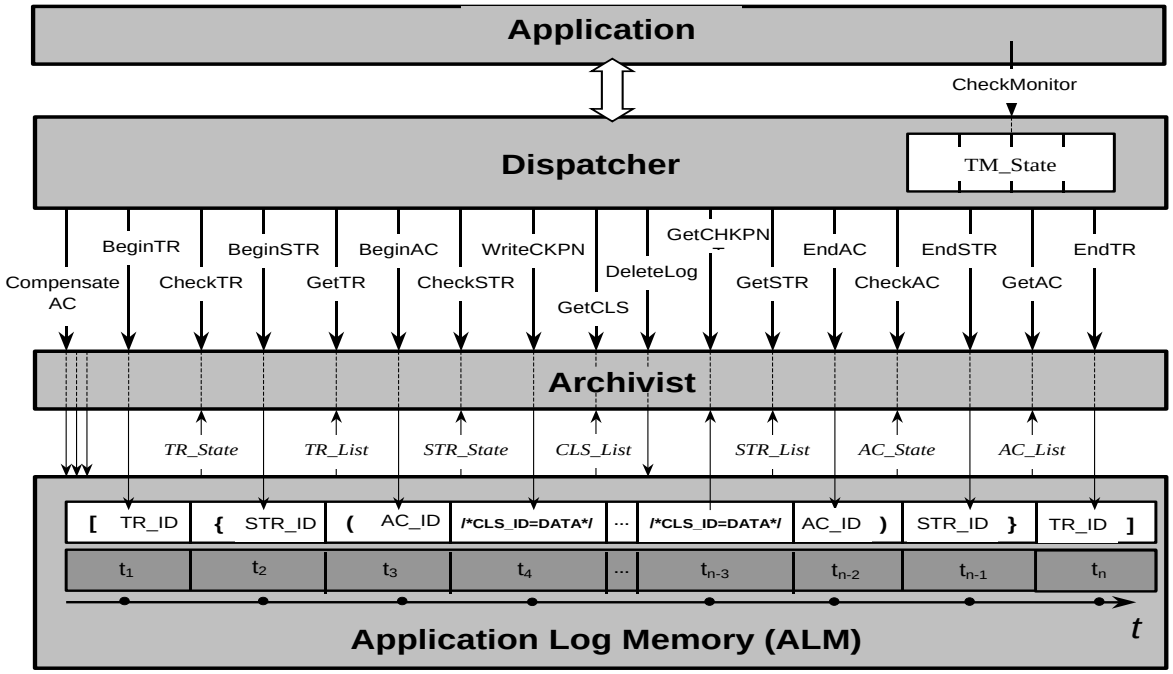


Fig. 3. Managing the Application Log Memory

Operations GetTR, GetSTR, GetAC, and GetCLS are used by the Application to pick up from the ALM Ids of transactions within temporal interval, Ids of subtransactions within a transaction, Ids of actions within a subtransaction, Ids of data clusters within an action. Operations CheckTR, CheckSTR, and CheckAC return results of analysis of a transaction, a subtransaction, or an action, respectively, to allow Application to decide what to do with previously interrupted transaction (subtransaction or action). Operation DeleteLog purges the ALM. Through a parameter the application can ask the TM to keep a data cluster in the persistent log for a later usage (e.g. a digital map). Operation CompensateAC marks that the action to be logged compensates an earlier action. This helps in guaranteeing idempotence in case a crash occurs during the recovery from earlier crashes.

No item is ever cleaned from the log, except when the whole ALM for a particular transaction is discarded.

4.3. Simple MCTM Implementation

The TM sketched above is implemented using Java. The idea is that the software is included into application during compile time and that the MCTM runs as part of each process (or thread). In other words, there can be several simultaneously running instances of the MCTM within one terminal. This raises the question, how the uniqueness of the TIDs is guaranteed. How this is achieved depends on the concrete environment. If the file system offers exclusive access to a file, this can be used to store a counter value into a special file. The MCTM reads this in an exclusive mode when allocating a new unique id and a higher counter value is written back to the file in exclusive mode. A more sophisticated solution is to use the local time provided by the clock of the terminal as part of the TID, with or without the counter above, which is the solution used in our implementation.

Another issue closely related with the architecture above is how to get the recovery started. MCTM can namely only run as part of an application process and another program must ask it to check the log(s). One solution to this is that after a terminal or application crash a special recovery application runs and asks the MCTM to return the list of all non-completed transactions (within a certain interval ending NOW). This list is then shown to the user who can select a TID. The recovery application subsequently retrieves the application type information from the ALM corresponding to the TID and runs the appropriate application up. The user can further ask the application to recover the transaction with TID. Another way to come to this point is that after a crash the user simply runs her interrupted application up. She can then ask the MCTM to list all incomplete transactions generated by that application type and pick one of them to be executed. Once the transaction to be continued has been chosen, the application can ask the MCTM to return its state to the application. It is the contents of the appropriate ALM, i.e. an ordered list of log entries. This includes the possible checkpoints. By following the list of BeginTR, EndTR, BeginSTR, EndSTR, BeginAC, EndAC and CompensateAC and checkpoints it can determine which

actions and subtransactions were completed, which not, and retrieve its internal state from the checkpoint file, if present. Depending on the state, the application must then either continue the execution (recover forward) or try to cancel the already taken actions at the servers (recover backward) - perhaps with user's help.

4.4. An application running with simple MCTM

The concrete application we are currently using with the MCTM is a Location-oriented application that offers to the user a digital map on PTD's screen based on the location the user is at. At the present stage the *Mobile Location Service (MLS) pilot system* [50] supports geographic data in the form of road network and location-based information on points of interest, encoded in XML. The MLS client runs on devices supporting Java such as Nokia 9210, laptops, and generic Symbian devices. A more detailed application-driven MCTM design can be found in [51] and the overall system design and implementation in [50]. The application is able to store the map content into a file, ask the MCTM to make a checkpoint and retrieve a checkpoint. When running up the application or during operation it can be asked to resume an incomplete transaction. In this case it reads the log and retrieves from the checkpointed file the contents of the map, if available. Otherwise, the execution can be continued from the point where it was interrupted. The current implementation does not support automatic backward recovery of the entire m-commerce transaction, should this be necessary. This must be done manually, based on the XML-encoded log that can be displayed by a browser or a text editor.

5. Ontological MCTM design

In the above simple MCTM design the application or the user has the understanding of the semantics of the workflow root actions, whereas the MCTM is only a slightly enhanced logger. The combination of security and privacy with transactional properties also fall short in the architecture. A more sophisticated MCTM would monitor the communications between the client and server and also integrate encryption/decryption functionality into the transactions. The encryption of messages protects the user from several threats that she can be exposed to, should the messages travel within the wireless and wired infrastructure unencrypted. These aspects are well understood in current computer networks and a suitable encryption measures are offered to protect both security and privacy of the user. In 2G and 3G telecom networks the traffic on the radio path is always encrypted. Less clear are the threats that are caused by the possibility of the mobile terminals used for m-commerce to be stolen. If the data logged during the enactment of m-commerce transactions and authorisation data is stored unencrypted into the terminal, a malicious person can read it from the stolen device and get hold of this private information, such as user-ids and passwords, perhaps also credit card number. Using these he or she might be able to launch new m-commerce transactions or other transactions on the cost of the legal owner. In some cases the privacy breach might be a big problem for the terminal owner.

As a new threat type in m-commerce we have earlier identified the possibility to "steal the root transactions" [39]. This happens when there are open m-commerce transactions when the terminal is stolen. The thief can then continue them in his or her interest, unless the user is protected against this threat. One way of protecting against this threat is to require user authentication as part of committing the purchases or changing delivery addresses. Other means to fight this threat is to encrypt the transaction logs and keep the authentication information either encrypted at the terminal or require it to be given explicitly by the user through keypad.

The architecture responding to the above requirements, in addition to those presented earlier is depicted in Fig.4.

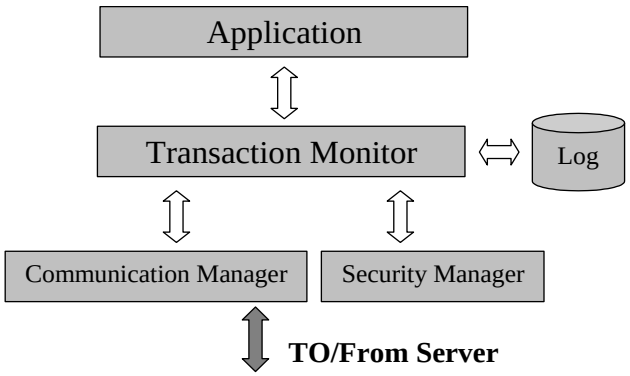


Fig. 4. Architecture of an Ontological TM

In this architecture, the application-MCTM interaction can happen at different abstraction levels. The generic low-level interface would be one where the application asks primarily the MCTM to communicate a request to a particular server and return a response. The request would be basically of form Req(IP-addr, PDU-content, Encr-indicator,TID). The MCTM then causes the encryption/decryption to be performed using the Security Manager that hosts the private key etc. and asks the Communication Manager to send the encrypted request to the server whose address is IP-addr, selected by the application. When the response arrives, MCTM decrypts it and hands the content to the application, along the TID. The MCTM can deduce the action borders and subtransaction borders by itself based on the server address in the request. The application must, however, indicate the beginning and end of the entire m-commerce transaction to the MCTM.

So, what is needed from the MCTM interface operation in Fig. 3 are operations that mark the entire transaction borders, checkpointing, and a command to retrieve the log of an incomplete transaction (CheckTR). Further, Rollback command should be added to the interface. Launching it by the application (or user) would cause the ontological MCTM to take the transaction log and run compensating subtransactions for those subtransactions that need compensation, and within them, compensating actions for individual actions. Should this not be possible for any reason, the MCTM should ask the application or user to resolve the problem.

Forward recovery of an application would be based on the CheckTR command that returns the log to the application. Analysing it, the application knows which actions were executed and can also retrieve its internal state, if it has stored a checkpoint. It can then continue execution from that state. The attempt to recover forward can also end with rollback, if this is the only reasonable alternative in the state retrieved.

The interface between the application and MCTM can be also more abstract than above. Whereas above the application must know the addresses of the servers it wants to communicate with and the application protocol format, these both could be catered for by the MCTM. In the extreme case, the application could only issue a request to the MCTM "Order taxi immediately to this place, target address being…" After that, the MCTM should interpret the request and compile or select a detailed transaction execution plan consisting of the steps presented in Section 2.3 above.

As can be seen, the latter requires a lot of intelligence from the ontological MCTM. In the most general case it should be able to generate a workflow (application) based on the request, in the least demanding setting it should be able to start a correct application based on the request. It is clear at this moment that ontologies must be used when generating the workflows from abstract requests, but what the limits for such an approach are, are currently unclear. These issues are investigated at the moment e.g. in EU-funded ASG project [49]. How transactions and transactional properties of m-commerce transactions can be supported by ontologies is currently a largely open research issue.

6. Related Work

The evolution of business environments towards mobile commerce requires that the use of mobile devices for conducting business transactions should not be a hard task for a user, even more - it needs to be made attractive. Launching m-commerce transactions and using mobile payment systems must be simple, reliable, secure and ubiquitous. Early mobile transactions required up to 60 key presses, whereas ideally this number needs to be fewer than five [44].As was discussed above in section 5, a new threat for security and privacy is that entire terminal will be stolen, in addition to the known threats in networked environments. The probability of this is evidently considerably larger than for stationary PCs or for laptops. Before users start making payments or sending their credit card details across the mobile Internet, they need to have faith that the appropriate security is in place in the network, but also at the terminal.

Important subproblems in m-commerce are payments and their standardised support as the examples above show; a payment subtransaction is a natural part in an m-commerce transaction. Although m-commerce transactions do not necessarily require mobile payment to pay for the goods or services ordered, mobile payment support makes m-commerce transactions more attractive for the customers. In fact, POS m-commerce transactions can consist solely of payment transaction if tangible goods are paid at a cash register using FeliCa® or other short-range wireless technologies between the terminal and the cash register. Mobile payments are especially important for the so-called micro-payments where the value of the goods is small and e.g. using a credit card would cost more than the goods or services themselves.

In Europe, the standardisation issue was targeted by launching Simpay in 2003, but the effort was stopped in June 2005 [70]. The idea was to provide Europe-wide end-to-end support for mobile payments using mobile terminals. The scheme was meant to support payments across multiple carrier networks, allowing users to pay for goods and services from a wide variety of retailers and merchants and collect payment data to their mobile phone bills. According to [70], the work continues now at the national level.

Another approach is so-called Mobile Wallet that allows users to pay for goods and services from their mobile devices using credit or debit cards. The Mobile Wallet idea has also been adopted by MeT [5]. In Japan, NTT DoCoMo launched

its i-mode™ FeliCa® service in July 2004 that uses FeliCa contactless IC chip technology developed by Sony Corporation for Mobile Wallet. By mid-May, 2005 it had sold 3.34 million FeliCa®compatible handsets [52]. By waving the handset above a suitable reader/writer, i-mode™ FeliCa system allows the customer to pay for goods and services. In the current development phase the customer can also get information of the products using the information capture function ToruCa™ [52] The handset can further be used as a house key or e-ticket. NTT DoCoMo is spreading this technology with new alliances in Japan and also abroad. [52].

Transaction support is crucial in mobile data management problems Specific characteristics of mobile environments (e.g. variable bandwidth, communication and execution autonomy of the terminals (cf. [10,15, Ch.12]), and limited resources on mobile terminals) make traditional transaction management techniques no longer appropriate. Serrano-Alvarado *et al* in their review [53] analyze and compare several contributions to mobile transactions. The analysis distinguishes two groups of models. The first group includes proposals where transactions are completely or partially executed on mobile hosts. In this group the focus is on ACID property support at the terminal. The second group considers transactions requested by mobile hosts and executed on the wired network nodes. In this case, ACID properties are not compromised and focus is on supporting mobile host movements during transaction execution.

Pervasive environments, unlike traditional mobile computing paradigm, do not differentiate between clients and servers that are located in a fixed, wired infrastructure, i.e. all the devices are modelled as peers. These environments also relax other assumptions made by mobile computing paradigm, such as the possibility of reconnection with a given device, or support from wired infrastructure. These fundamental characteristics of pervasive computing environments limit the use of techniques developed for transactions in "mobile" computing environments. Perich *et al* [54] examine the problem of transaction management in pervasive computing environments and present a new approach to address them. They represent each entity as a mobile or static semiautonomous device. The purpose of each device is to satisfy user queries based on its local data repository and interactions with other devices currently in its vicinity. An alternative optimistic transaction model is designed that is supposed to provide a high rate of successful transaction terminations and to maintain a neighbourhood-based consistency. The model accomplishes this via the help of active witnesses and by employing an epidemic voting protocol. The advantage of the model is claimed to be in enabling several peers in a pervasive environment to engage in a reliable and consistent transaction without assuming that they can communicate over a cellular network infrastructure. Transaction terminations in that case does not depend on any single point of a failure and appropriate protocol does not require all entities to be simultaneously connected, which provides communication autonomy for the terminals. It remains unclear, whether other node autonomy aspects are supported.

Varshney and Ravikumar [55] focus on supporting transactions for group-oriented m-commerce services. They analyze and characterize the transaction requirements, diverse transactions protocols and their suitability for group-oriented m-commerce services. Their results show that the three suggested protocols could support a range of performance requirements, including varying probability for a transaction completion and transaction delay, for group-oriented m-commerce services. The transaction delays depend on a user connectivity/disconnectivity ratio and average duration, and the transaction blocking mainly depends on the average duration of disconnection. It is also shown that increased application flexibility, due to a reduced number of entities needed for a transaction completion, can compensate the negative impact of frequent user disconnections by improving transaction delay and blocking performance.

Comparing these results to ours, one sees that we did not analyze explicitly the disconnection problematic. In the simple MCTM case it is the problem of the appliction/user to recover from the communication failure. If the communication failure causes the application to crash or the user lets it crash, the state of the transaction can be retrieved from the log and forward or backward recovery performed. In case of the Ontological TM these tasks are performed by it. The performance results presented in [55] are not necessarily relevant for our case, because we assume a more complicated recovery strategy.

According to [55] the areas of mobile services where reliable transaction management is an important issue are as follows:

- Mobile Auctions [56].
- Mobile Multi-Party Interactive Games [57]
- Mobile Financial Services [58].
- Mobile and Location-Based Advertising [59].
- Mobile Product Recommender Systems [60].
- Mobile Patient Monitoring [61].
- Mobile Entertainment Services [62]
- Mobile Distance Education [63]
- Proactive Service Management [64].

Many of these applications or services are not directly related even with M-business, and even less with m-commerce. There can still be structural similarities in the execution structures of these distributed computations and m-commerce transactions, as well as similar assumption of the behavior of the nodes. These are for further study.

Emerging Semantic Web standards allow semantic modelling of various domains to ensure interoperability among heterogeneous Web applications. The transaction management domain is not an exception. For example in [65] a simple ontological model has been considered for a mobile electronic commerce case. The approach is based on the assumption that the transaction management tool can be implemented in a mobile terminal, allowing integration of different distributed external Web-services. The transactions were managed across multiple location-aware Web-services. Ontologies are supposed to be placed (or accessed) both in mobile terminals and in Web services. They define common standards and vocabularies for the use of the names, types, schemas, default values for parameters, atomic service components with appropriate structure of component's profile, input and output.

Support for transactional behaviour and fault tolerant execution of mobile agent-based applications is also a fundamental issue in the development of mobile agent systems. In [66] the MARES platform was presented, which supports the modelling of mobile agent-based applications as distributed transactions. The executions of mobile agent-based transactions on top of the platform are fault-tolerant, so that if the location where a part of a global transaction is being executed becomes faulty long enough, the system performs a recovery procedure to resume the execution of that part of the transaction at another location.

Pervasive environments are characterized by a variety of wireless devices ranging from sensors to mobile phones and laptops. P2P information sharing in such environments presents a unique opportunity for people and devices to exchange information such as documents, music, pictures, movies, etc with peers. Thus, information in pervasive environments, which has a monetary value, introduces a new set of problems for P2P interaction. In [67], a middleware has been discussed that attempts to solve these problems using concepts of the Contract Net Protocol, Semantic Service Discovery and secure transaction protocols. It was assumed that the ontology (in DAML+OIL) is shared in P2P pervasive environment so that peers can "understand" meaning of each other messages. A Prolog-based reasoning engine is used to discover match between incoming and locally stored attributes. The new generation of mobile phones is expected to enable various objects to be distributed across a range of devices. Such object sharing and mobility will provide support for rich client interaction. In these environments, mobile devices are under the control of individuals, not under the direct control of the organisation, and the mobile devices exhibit therefore execution and communication autonomy [10,15;Ch.12] towards the infrastructure (servers, network) and towards other mobile devices. As is known since the beginning of 1990'ies, enforcing ACID properties is in contradiction with various aspects of node autonomy [10,15]. The solution to preserve autonomy is to allow more concurrency among global subtransactions running at different nodes than what the syntactical serializability (ACID properties for RW-level) allows. If the sites are using locking, the locks must be released when a global subtransaction ends, i.e. before the global commit or abort of the root transaction is performed. This leads to the problem that other local and global transactions can access the released data and also modify the data items, even when the global subtransaction should be rolled-back (compensated). This means dirty reads and should lead to cascading aborts [12], but the committed transactions cannot be aborted any more. The only way out is to use compensation for failing global transactions. In [68] the issues of distributed architectures are discussed where objects are able to run on mobile devices, but must be universally synchronized. The paper recognizes the importance of a transactional mechanism in supporting synchronization. Problems with locking are also referred to, but the issues are not properly related with autonomy concepts.

On-line business transaction processing systems have so far been based on centralized or client-server architectures. In [69] it is claimed that such systems may also be based on the constantly evolving decentralized peer-to-peer architectures. As part of the functionality of these, a suitable transactional support for various system functionalities (support for workflow and collaboration orchestration, transaction phases, logging, non-repudiation; non-functional properties, including security, availability, anonymity) should be provided. The authors conclude that current peer-to-peer technology has evolved to the extent that it is able to fulfil many of these requirements to a large extent.

From the brief review above the trend can be seen of implementing emerging technologies and standards to target the transaction management issues in mobile commerce. These include W3C standards on Semantic Web, metadata and ontologies, agent, multiagent and mobile agent technologies, decentralized peer-to-peer architectures, various AI methods and reasoning tools, etc. We believe that smart integration of various, emerging and heterogeneous approaches and tools may provide a good basis for optimism concerning the future of mobile commerce transactions.

7. Conclusions

Currently one can see the emergence of at least five different kinds of m-commerce transaction types: Internet e-commerce over wireless access networks, location-based services, ticketing applications, retail shopping, and banking. M-commerce operates partially in a different environment than Internet E-Commerce due to the special characteristics and constraints of the terminals, sometimes called Personal Trusted Devices, and networks, and the context, situations and circumstances that people use their PTDs while roaming.

In this paper, we have analysed the business models and ensuing transactional requirements in the environment and deduce key ingredients for a transaction model necessary for m-commerce environments.

We concentrated mostly on the two first types of m-commerce transactions mentioned above in this paper, namely mobile Internet e-commerce and location-based services. Central conclusions are that m-commerce transactions in a strong sense are needed. The key insight is that m-commerce transactions should be seen as distributed workflows with transactional properties. Their modelling can be based on RDAGs and their properties expressed using these. It is evident from the analysis of the m-commerce environment that only the first level below the root transaction is controllable by the root. The deeper levels in the execution hierarchy are dictated by the business relationships between the business actors and can neither be determined nor controlled by the customer launching the m-commerce transaction on her terminal. M-commerce transactions as transactional workflows are relatives of S-transactions, i.e. structurally RDAGs, long lasting, contain cancellations and real actions. A form of semantic atomicity (preferably certified delivery [28]) is the property they try to enforce.

Another important conclusion is that the security and privacy issues must be combined in a different way with the transaction modelling concepts than before. This is mainly because the devices used to launch m-commerce transactions can be stolen by malicious people and thus the transaction root "hijacked" by them. Another issue related with loosing the device is that privacy can be jeopardized due to the information contained in the transaction logs or elsewhere within the device.

The m-commerce environment is global, highly autonomous and heterogeneous due to roaming, different regulatory frameworks and existing and emerging business models. This causes roaming heterogeneity, a new form of heterogeneity, to emerge. That and other forms of heterogeneity make it worthwhile to develop standard solutions to reduce heterogeneity and make global m-commerce possible, an important task for Open Mobile Alliance. As the experience with Symbay shows, this is not always easy.

An important conclusion is that the m-commerce transactions are important both conceptually and also pragmatically, especially when the mobile commerce revenues grow. Although at least to certain extent coherent view on them has been presented here, rendering e.g. standard software at the terminal feasible, much more detailed work is needed. Especially the marriage of Semantic Web and transaction concepts requires a lot of work, but promises interesting results. The formal m-commerce transaction model with its typical properties is for further study.

Acknowledgements

This research was funded by the Finnish National Technology Agency (TEKES), Nokia Networks, HP Finland, and Yomi Solutions (contract 330/401/99). Support of FhG-FIT (Sankt Augustin, Germany) for the first author during his sabbatical in 2000-2001 is also highly appreciated.

References

[1] A. Helal, B. Haskell, J.L. Carter, R. Brice, D. Woelk, M. Rusinkiewicz, Any Time, Anywhere Computing; Mobile Computing Concepts and Technology, Kluwer Academic Publishers, June 1999.

[2] K. Tanaka, A set of transparencies about the Finnish-Japanese 3G project, Finpro Office, Tokyo, 2001.

[3] K. Ichikawa, The View of NTT DoCoMo on the Further development of Wireless Internet. Tokyo Mobile Round Table Conference, May 2002 (CD).

[4] New Media Review. Accessed March 30, 2005, at http://www.etcnewmedia.com/review/default.asp?SectionID=10&OverviewID=6

[5] MeT White Paper on Mobile Transactions, 22.1.2003. Accessed on March 30, 2005 at http://www.mobiletransaction.org/pdf/R200/white_papers/MeT_White_paper_on_mobile_transactions_v1.pdf. MeT Overview White Paper. Version 2.0, 29.1.2001. http: //www.mobiletransaction.org/pdf/White%20Paper_2.0.pdf.

[6] Measuring the Information Economy 2002, Annex 4 "The OECD definitions of Internet and e-commerce transactions". OECD 2002. Accessed Sept. 12, 2005 at http://www.oecd.org/dataoecd/34/16/2771174.pdf.

[7] R.Kalakota, N.Robinson, M-business: The Race to Mobility. McGraw-Hill Companies, New York, USA, 2002.

[8] K.Turowski, K.Pousttchi, Mobile Commerce, Grundlagen und Techniken, Springer, Heidelberg, Germany, 2004.

[9] J. Funk, private communication.
[10] J. Veijalainen, Transaction Concepts in Autonomous Database Environments. (Ph.D. thesis). GMD-Bericht Nr. 183, R. Oldenbourg Verlag, Munich, Germany, April 1990.
[11] J. Veijalainen, A. Wolski, Transaction-based Recovery. Chapter 11 in: A. Elmagarmid, M. Rusinkiewicz and A. Sheth (eds.), Management of Heterogeneous and Autonomous Database Systems, Morgan Kaufmann Publishers. San Francisco, October 1998, pp. 301-350.
[12] P. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.
[13] J. Gray, A. Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann, San Mateo, CA, 1993.
[14] C. Papadimitriou, The Theory of Concurrency Control, CS Press, Rockville, MD, 1986.
[15] A. Elmagarmid, Database Transaction Models for Advanced Applications, Morgan Kaufmann, 1992.
[16] R. De By, W. Klas, J. Veijalainen (eds.), Transaction Management Support for Cooperative Applications. Kluwer Academic Publ., December 1997.
[17] MeT, Mobile Electronic Transactions forum. http://www.mobiletransaction.org/
[18] Open Mobile Alliance. Accessed March 30, 2005 at http://www.openmobilealliance.org/
[19] P. Bernstein et al. The Asilomar Report on Database Research. SIGMOD Record 27 (1998), 4(Dec.). http:www.acm.org/sigmod/record/issues/9812/asilomar.html.
[20] J. Veijalainen, Transactions in Mobile Electronic Commerce. In: G. Saake, K. Schwarz, C. Türker (eds.), Transactions and Database Dynamics. LNCS Nr. 1773, Springer Verlag, Berlin, December 1999: 208- 229.
[21] O. Braun, D. Bremer, G. Schmidt, K. Zimmer, Designing a Generic System for Process-Oriented Support of Business Transactions Using Internet for Electronic Commerce, http://www.electronicmarkets.org/netacademy/.
[22] A. Turner, Internet Contributes to Increase in Identity Theft, Fairfax IT, September 1, 2000, available in: http://www.it.fairfax.com.au/breaking/20000901/A41152-2000Sep1.html#top.
[23] J. Veijalainen, F. Eliassen, B. Holtkamp: The S-transaction Model. Chapter 12 in [15]
[24] H. Garcia-Molina, K. Salem: Sagas. Proc. of ACM SIGMOD conference, May 1987, pp. 249-259.
[25] H. Garcia-Molina, D. Gawlick, J. Klein, K. Kleissner, K. Salem, *Modeling Long-Running Activities as Nested Sagas*, IEEE Bulletin of the Technical Committee on Data Engineering, 14 (1), 1991.
[26] H. Garcia-Molina et al. Coordinating Multitransaction Activities with Nested Sagas. Ch. 16 in V. Kumar & M. Hsu (eds.), Recovery Mechanisms in Database Systems, Prentice-Hall 1998.
[27] P. Grefen, J. Vonk, P. Apers, Global Transaction Support for Workflow Management Systems: From Formal Specification to Practical Implementation. VLDB Journal 10 (2001), pp. 316-333.
[28] J. D. Tygar, Atomicity in Electronic Commerce. In Proceedings of the 15th PODC Conference, 1996: 8-26.
[29] J. Veijalainen, M. Weske, Modeling Static Aspects of Mobile Electronic Commerce Environments. Ch. 7. in L. Peng, K. Siau (eds.). Advances in Mobile Commerce Technologies, Idea Group Publishing, 2003.
[30] P. Timmers, Business Models for Electronic Markets. http://www.electronicmarkets.org/netacademy/publications.nsf/all_pk/949/$file/v8n2_timmers.pdf?OpenElement&id=949.
[31] A. Devine, S. Holmqvist, Mobile Internet Content Providers and their Business Models. Masters Thesis, Stockholm Kungl Tekniska Högskolan, January 2001. Accessed Sept. 13, 2005 at http://www.japaninc.net/online/sc/master_thesis_as1.pdf.
[32] J.L.Funk, Mobile Disruption, The Technologies and Applications Driving the Mobile Internet. John Wiley&Sons, New Jersey, USA, 2004.
[33] P. Tarasewitch, M. Warkentin, Issues in Wireless E-Commerce, ACM SIGEcom Exchanges, Issue 1.1, August 2000, pp. 19-25.
[34] WAP forum; WAP Specification 2.0. Accessed on Sept. 13, 2005 at http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html.
[35] Nordea Solo. Accessed Sept 15, 2005 at http://solo.nordea.fi.
[36] Nordea bank. Accessed Sept. 15, 2005 at http://www.nordea.com/
[37] Keltainen Pörssi. Accessed Sept. 15, 2005 at www.keltainenporssi.fi.
[38] Liquidaudio. Accessed Sept. 15, 2005 at http://www.liquidaudio.com/
[39] J. Veijalainen, A.Tsalgatidou, Electronic Commerce Transactions in a Mobile Computing Environment. In: Q.Jin, J. Li, N. Zhang, J. Cheng, C. Yu, S. Noguchi (eds), Enabling Society with Information Technology,. Springer Verlag, Tokyo, 2002, pp. 131-140.
[40] Nordea Solo over WAP services. Accessed Sept. 13, 2005 at http://www.nordea.fi/sitemod/default/index.aspx?pid=700654.
[41] G. Letham, GIS Movers and Shakers Target LBS. http://spa tialnews.geocomm.com/newsletter/2000/22/mobile2001l.
[42] J. Suutari, Transparencies on the Architecture of Location-based Services in WAP Environment, MMM-Nokia Meeting, November 2000.
[43] J. Tang, A. W. Fu, and J.Veijalainen: Supporting Dispute Handling in E-Commerce Transactions, a Framework and Related Methodologies. Electronic Commerce Research, 4 (4) (2004) 393–413.
[44] S. Wallage, Mobile Payments Key to Rise in M-Commerce, 3 GSM World Congress 2005. Accessed on Sept. 15,, 2005 at: http://www. businessweek.com/adsections/2005/pdf/0508_3gsm.pdf.
[45] J. Veijalainen, M. Weske, Modeling Static Aspects of Mobile Electronic Commerce Environments. Chapter 7 in "Advances in Mobile Commerce Technologies", Lim Ee Peng, Keng Siau (eds.). IDEA group publishing, 2003, pp.137-170.

[46] D. E. Denning, P. F. MacDoran, Location-Based Authentication: Grounding Cyberspace for Better Security, Computer Fraud & Security, Elsevier Science Ltd., 1996, http://www.cosc.georgetown.edu/~denning/infosec/Grounding.txt.

[47] C. Herring and Z. Milosevic. Implementing B2B Contracts Using BizTalk. Proc. of HICSS-34, Maui, Hawaii, Jan. 3-6, 2001. CD-ROM, file ST4T106.pdf.

[48] Web Service Modeling Ontology (WSMO). Accessed Sept. 11, 2005 at http://www.wsmo.org.

[49] Adaptive Services Grid (ASG), EU FP6 Project 004617. Accessed Sept. 11, 2005 at http://asg-platform.org.

[50] J. Markkula, A. Katasonov, A. Garmash, Developing MLS Location-based Service Pilot System. In: Proc. of Smartnet'2002, 7th Conference on Intelligence in Networks, Saariselkä, Finland, April 8-10, 2002.

[51] J. Veijalainen, V. Terziyan, Transaction Management for M-Commerce at a Mobile Terminal. Accepted to Workshop on Reliable and Secure Applications in Mobile Environment. Oct. 28, New Orleans, USA. http://www.cs.jyu.fi/~mmm

[52] i-mode™ FeliCa® of NTT DoCoMo, Japan. Accessed Sept. 11, 2005 at http://www.nttdocomo.com.

[53] P. Serrano-Alvarado, C. Roncancio and M. Adiba, A Survey of Mobile Transactions, In: *Distributed and Parallel Databases*, Vol. 16, No. 2, September 2004, Kluwer Academic Publishers, pp. 193-230.

[54] F. Perich, A. Joshi, Y. Yesha, and T. Finin, Neighborhood-Consistent Transaction Management for Pervasive Computing Environments, In: *Proceedings of the 14th International Conference on Database and Expert Systems Applications*, LNCS 2736, Springer, 2003, pp. 276-286.

[55] U. Varshney, and K. Ravikumar, Transaction and Networking Support for Group-Oriented Mobile Commerce Services, In: *Proceedings of the* IEEE/ACM First International Workshop on Broadband Wireless Services and Applications (BroadWISE 04), 2004, IEEE CS Press.

[56] J. Sairamesh, S. Goh, I. Stanoi, C. S. Li, and S. Padmanabhan, "Selfmanaging, Disconnected Processes and Mechanisms for Mobile Ebusiness", In Proc. 2nd Int. Workshop on Mobile Commerce, pp.82-89, 2002.

[57] J.-P. Partanen, Mobile Gaming: A Framework for Evaluating the Industry 2000-2005, Gaptime Century, 2001, available in: http://www.gaptime.com/mobilegaming.pdf

[58] U. Varshney, "Mobile Payments", IEEE Computer, vol. 35, no. 12, December 2002, pp. 120-121.

[59] U. Varshney "Location Management for Mobile Commerce Applications in Wireless Internet", ACM Transactions on Internet Technologies, vol. 3, no. 3, 2003, pp. 236-255.

[60] B.N. Miller, J. Konstan, J. Riedl, PocketLens: Toward a Personal Recommender System, *ACM Transactions on Information Systems (TOIS)*, Vol. 22, Issue 3 (July 2004), pp. 437-476.

[61] U. Varshney, "Patient Monitoring using Wireless Networks", in Proc. Of Americas Conference on Information Systems (AMCIS), August 2004.

[62] P. Kymäläinen (ed.), Mobile Entertainment Industry and Culture, IST-2001-38846 Project Deliverable D8.2.3 Mobile Entertainment, April 2004, available in: http://www. mgain.org/mgain-wp8-D823_book-delivered.pdf .

[63] J. Nogami, M. Kusanagi, R.B. Thapa, Tele-education experiment in the field of Remote Sensing and GIS, available in: http://www.gisdevelopment.net/application/ miscellaneous/ma03192.htm

[64] U. Varshney, and R. Vetter, "Framework, Applications, and Networking Support for M-commerce", ACM/Kluwer Journal on Mobile Networks and Applications (MONET), vol. 7, no. 3, June 2002, pp. 185-198.

[65] V. Terziyan, Ontological Modelling of E-Services to Ensure Appropriate Mobile Transactions, In: *International Journal of Intelligent Systems in Accounting, Finance and Management*, J. Wiley & Sons, Vol. 11, No.3, 2002, pp. 159-172.

[66] F. Silva, R. Macedo, A. Freitas, The MARES Platform: Support for Transactional and Fault-Tolerant Execution of Mobile Agent-based Applications, In Proceedings of the Fourth Workshop on Tests and Fault Tolerance, IVWTF 2003, co-located with the XXI Brazilian Symposium on Computer Networks, SBRC2003, Natal, RN, May 2003.

[67] S. Avancha, P. D'Souza, F. Perich, A. Joshi, and Y. Yesha, "P2P M-Commerce in Pervasive Environments", ACM SIGecom Exchanges: Special Issue on Mobile Commerce, Vol. 3, No. 4, pp. 1-9, January 2003.

[68] D. Parsons, Java Architectures for Mobilized Enterprise Systems, In: Proceedings of the 38th Hawaii International Conference on System Sciences, 2005.

[69] S. Androutsellis, D. Spinellis, Performing Peer-to-Peer E-Business Transactions: A Requirement Analysis and Preliminary Design Proposal, In: Proceedings of the IADIS e-Commerce 2004 Conference, Lisbon, Portugal, 2004.

[70] L. Sherriff, Simpay halts mobile commerce project, The Register, June 27, 2005. Accessed on Sept. 14, 2005 at http://www.theregister.co.uk/2005/06/27/simpay_halts_project/.