# Transaction Management for M-Commerce at a Mobile Terminal

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

*[1]Department of Computer Science and Inf. 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), very little attention has been paid for distributed transactional properties of the computations facilitating m-commerce. In this paper we first present a requirement analysis and then present a wireless terminal-based Transaction Manager (TM) architecture. This architecture is based on the assumption that there is an application that supports certain business transaction(s) and that it uses the TM to store transactional state information and retrieve it after a communication link, application, or terminal crash. We present the design of such a TM, including the application interface, modules and log structure. A pilot implementation of this TM for the location-based application is also discussed. We further discuss other alternatives to design such a TM that together can be called "Ontological Transaction Monitor". This acts as an intelligent component between the application and the servers accessed during M-commerce transactions and controls the perceivable communication behavior of the terminal towards the servers, maintains the state information and takes care of tight coupling of transactional properties of the computations as well as of security and privacy.*

## 1. Introduction

The main driving force for the rapid acceptance rate 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 [7]. 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 [16,38]. The telecom industry estimates that there will be 500 million Internet-enabled mobile phones in 2003 in the world. The number of these mobile Internet-enabled terminals, sometimes called Personal Trusted Devices (PTDs), is expected to exceed the number of fixed-line Internet users around 2003 [12]. M-commerce transactions are an important class of applications on the PTDs. Thus, it is of high importance that the infrastructure offers proper security and transactional means to protect all players in the environment against system crashes, but also against malicious players and criminals.

It is not very clear yet what are transactions in m-commerce context and what could be their exact transactional properties. The term “transaction” had almost ten closely related, but different meanings, ranging from business transactions to formal model of program execution within a database system [21,24]. So far, most of the transaction modeling work has been done from the database system perspective. The famous (but imprecise) ACID properties should be guaranteed for the program executions within the database system [1,14,6]. The more complicated models present individual transactional computations as trees. One modeling dimension is the selection of the correctness criteria that divide the histories into acceptable and non-acceptable ones. A standard way of doing this is to set up an equivalence relation among histories and classify them with a serial history as serializable, i.e. acceptable. Check e.g. [3,24] for a more complete analysis of diverse transaction modeling incentive.

Mobile e-commerce transactions have been developed in an industry-led consortium called MeT-forum [11]. The work has produced a public white paper [12] where the opportunities and risks of m-commerce are discussed. Scenarios (business models) for five types of m-commerce have been developed. On June 12th, 2002, a larger consortium called Open Mobile Software Alliance (OMA) was announced [36]. 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 be considered. MeT consortium will join this initiative.

Is m-commerce an area that would again need its own transaction model? Isn’t the work of MeT enough? In a closer look the need to go beyond individual messages and message exchanges becomes more than evident, because the overall business transaction can consist of several interactions with different players, 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 paid, or goods 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 [37].

Another issue is the security and trust closely related with it. 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 whole [25]. Risks are discussed e.g. in [2,19].

A tentative definition for m-commerce transactions can be stated as follows: a mobile e-commerce transaction is any type of business *transaction of an economic value that is conducted using a mobile terminal that communicates over a wireless telecommunications or Personal Area Network with the e-commerce infrastructure.*

M-commerce transactions are inherently distributed, because they are always performed over a wireless link and are thus protocol-driven. From the modeling point of view they can be viewed as special kind of workflow. M-commerce transaction refers to:

- a specification of a m-commerce workflow composed of a specification;
- 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 [1,14]. The same is true also for many “advanced” transaction models developed ([5,3,24]), 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 [21,23] developed for international banking environment with strong autonomy properties. Sagas [39] and nested sagas [40,41] are also of relevance, but as a special case of S-transactions they do need to be paid a special attention to. The most relevant work is reported in [42] where the workflow specification, as well as the transaction specification and execution graph, are closely tied together. We analyze below more closely the commonalties and differences of m-commerce workflow and those in [42].

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. As special e-commerce transactions, m-commerce transactions complying with the model should guarantee the atomicity notions introduced in [20] for e-commerce, which as such can be formulated as special kind of semantic constraints between subtransactions of S-transactions (cf. [21,23]).

In the sequel we deepen the above analysis about the need and form of transaction modeling for m-commerce environments. In section 2 we relate business models and transaction modeling concepts with each other. This is done by analyzing two of the five business scenario types suggested by MeT [12]. Based on this analysis we refine transactional requirements and properties, as well 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 a more sophisticated design. Section 6 concludes.

## 2. Business models for m-commerce

In our earlier work [43] 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 [18]:

- 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 players

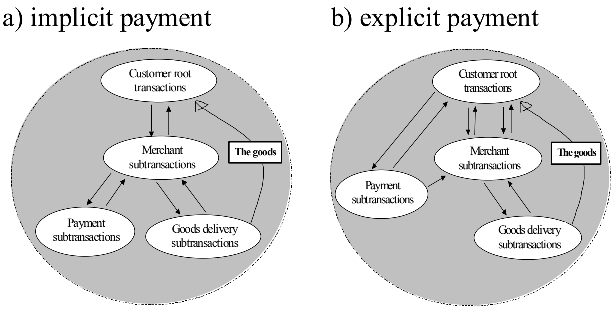
We first look at Japan, because the wireless Internet (e.g. i-Mode) has existed there since February 1999 and certain m-commerce player categories have emerged and are economically viable. According to [16,32] in the market initiated by NTT DoCoMo one can distinguish the following players: a user; mobile network operator (MNO); telecom operator; application provider; facility supplier; information provider; contents holder; solution provider; financial institution; terminal manufacturer. Not all these players 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 is that services are billed and thus payments are not performed on-line [32]. Thus, conventional telecom business models are adopted, rather than those applied in Internet environment.

## 2.2. Internet e-commerce over PTDs

The basic idea here is that the wireless network is one of the access networks to the e-commerce infrastructure offered over 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 [17]). 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 [18] 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 WML [27] or c-HTML – or XML-related markup languages, especially XHTML [35]. 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.

To give concrete examples we look at three business transactions in [18]. 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 [9,13,28]).

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

Customer root transactions
Merchant subtransactions
The goods
Payment subtransactions
Goods delivery subtransactions
Customer root transactions
Merchant subtransactions
The goods
Payment subtransactions
Goods delivery subtransactions

**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 [29]. 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 [22]), 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 PINs 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 [22].

In the banking case there are only two explicit players, the customer and the bank, such as Nordea [9,13]. 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 [10]. The interaction patterns remain the same as above, but the forms exchanged between the client and 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 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.

On the other hand, as was inferred in [25], 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. This requirement is evidently in contradiction with the above minimisation requirement.

### 2.3. Location-based services; the taxi example

These services are more in the domain of the MNOs than the previous ones [8]. 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?" A very useful service request is: "Send me a taxi right now!" For discussion on the emerging applications see [15].

The first and very basic query that is implicit in any location-based activity can be answered by displaying the WGS-84 coordinates, or using maps, voice, etc. 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)
positioning infra-> terminal: (MyId,XYZ)]
     Terminal: form the queryQ(MyId, XYZ,id,otherPara)
[Terminal ->LBS appl.:query
                                Q(MyId,XYZ,Qid,otherPara)
LBS application -> terminal
                    Result(Q(MyId,XYZ,Qid,otherPara))
{Terminal -> LBS application: ackQ(Qid)} ]
```

By "[", "]" we denote here the borders of subtransactions.

Let us now look at the taxi-ordering example. It differs from the above digital content delivering examples in that a physical taxi comes to the location where the customer is (or was when the taxi was ordered). We can use either of the ways above to perform the location query *Q*. Let us take the former form. Then we have:

```
[Terminal-> positioning infra: locate_me (MyId,Qid)
positioning infra-> terminal: Result( Qid,XYZ)]
 [Terminal: form the query Q(MyId, XYZ,id,otherPara)
Terminal ->LBS appl.:queryQ(MyId,XYZ,Qid,otherPara)
LBS application -> terminal
          Result(Q(MyId,XYZ,Qid,otherPara))
Terminal -> LBS application: ackQ(Qid)]
Taxi arrives to the place (XYZ) after W minutes from placing the
order.
Customer is transported to the target address
[Customer pays the trip]
```

In this case the Result(Q(MyId,XYZ,Qid,otherPara)) says "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 NackQ cancels the order. OtherPara 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. If W above 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 [30] 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.

Another aspect related with this is to map the (XYZ) coordinates to <address>. It requires most probably coordinate transformations and attachment of them to the local map. Which player provides this mapping? It can be the taxi ordering company, each individual taxi owner (GPS car map), or an external service provider that upon getting a coordinates <XYZ> returns the <address>. These infrastructure issues are for further study. The above taxi service requires privacy protection measures and customer trust. 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 with the customer. In general, m-commerce transaction security can probably be improved through location-based authentication (see [4]). 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 slow and expensive, the capabilities of the PTDs are limited, one is easily lead to the following idea. Another component, *proxy,* could present the terminal and thus the customer against other players. 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. From the m-commerce workflow point of view 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 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 be based on XML as BizTalk™ [31] is. This kind of a *mobile assignment/ contracting language* is left for further study.

## 3. Towards m-commerce transaction model

From the above analysis one can draw the following conclusions. In all m-commerce workflow specifications the user is the ultimate source and initiator of the workflow. Thus, the terminal should run in all cases the root step of the workflow instance. It is, however, as evident that there is not just one m-commerce workflow that the terminal should be able to run, but many, as evidenced by the few examples above. They differ in topology from each other. In addition, different workflows might require different kinds of protocols to be run against the other players. Taking into consideration the local business habits, even the "same" abstract workflow, such as "ordering a taxi" – might require different concrete protocols to be run at the terminal in different countries. We can still find many commonalties between the protocols and a common shape of the execution graphs. Thus, as alluded in the introduction, *m-commerce transaction model* is a reasonable concept. What are the properties of such transactions?

The PTD representing the customer is the source of activity and thus naturally the *root* of the transaction. It must often request several tasks or steps to be performed at different players (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. [42]). There are also contractor-subcontractor relationships present in the environment (logistics, searching, indirect payments) that necessitate a deeper *subtransaction hierarchy.* Akin to S-transactions the depth of the hierarchy cannot be determined by the root at the beginning of the execution. 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 are already existing ones 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 *local* 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 [12] and later OMA [36]. The roaming heterogeneity and existing global infrastructure is a difference to S-transactions as well as to the environment in [42]. In order to mitigate the above problem and to cut the data transmission costs one must try to minimise the number of parties 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.

Like S-transactions, M-commerce transactions often contain *real actions* [6,21] 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 real world action (like arriving of the ordered book).

Similarly to S-transactions, *atomicity preservation* is the key property of the m-commerce transactions. That is, they always try to enforce an *atomicity constraint* [21] 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 [15], 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.

Durability, Consistency and Serializability are important concerning the subtransactions of different m-commerce transactions at one player, but they are not the key properties of the overall transaction. In fact, they play very similar role as in S-transactions [21,23]. *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 [22,25]. 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. 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. Could workflow specifications be made executable at PTDs? In theory yes, but in practice this would require much resources 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 9210 and 3G terminals in Japan) are now capable of running Java so it is only a matter of time when this will 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. 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 threats? 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 is not 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 above 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 identity of the player, where 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 can 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 properties 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(<id>)* and *resume(<id>)* 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: 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. This requires at least a persistent log from which the history prior to the crash can be retrieved. We opt for a design where the TM is application-driven and acts as a kind of sophisticated persistent memory and bookkeeper. From the customer point of view, it is reasonable to keep record of not only incomplete transactions, but also of finalized ones. There should be possibility to save transaction log or part of it into an “application” log upon transaction termination.

### 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.2. Application Interface is implemented by Dispatcher that keeps track of the TIDs etc. and is able also to analyze the state of the transaction. 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 stage 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.

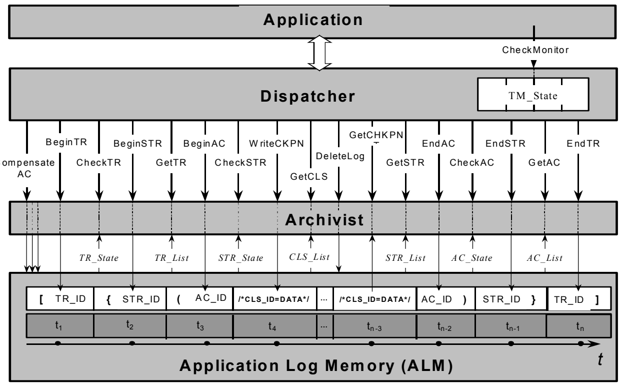


**Fig. 2.** Managing the Application Log Memory

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* [34] 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. A more detailed application-driven MCTM design can be found in [33] and the overall system design and implementation in [34].

## 5. Ontological MCTM design

In the above simple MCTM design the application has the understanding of the semantics of the workflow root actions, whereas the MCTM is only a slightly enhanced logger. The combination of security, privacy and 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 architecture is depicted in Fig.3:

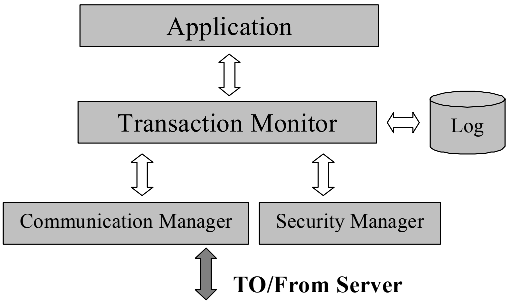


**Fig. 3**. Architecture of an Ontological TM

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. 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 a detailed transaction execution plan consisting of the following steps: (1) ask the positioning of the terminal; (2) consult the service discovery service in order to find the most appropriate taxi service address in the neighbourhood; (3) send request to this service for a taxi; (4) inform the user that a taxi is coming along the order information.

As can be seen, the latter requires a lot of intelligence of the TM. It should be able to generate an application based on the request. It is worth of noticing that the Ontological TM could use the earlier application driven MCTM as its component.

# 6. Conclusions

Currently one can see the emergence of at least five different kinds of m-commerce applications: 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. Central conclusions are that m-commerce transactions in a strong sense are needed. We have also implemented the first manager running at Java-enabled terminals, such as Nokia 9210. M-commerce transactions can be viewed as transactional workflows and are relatives of S-transactions, i.e. structurally RDAGs, long lasting, contain cancellations and real actions. A form of semantic atomicity (preferably certified delivery [20]) is the property they try to enforce. Due to hostility of the environment, and vulnerability of PTDs, security and privacy must be intertwined with the transaction model and its realisation. 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. An important conclusion is that the m-commerce transactions are an important conceptually and pragmatically. Although a rather coherent view on them has been presented here, making standard software at the terminal feasible, much more detailed work is needed.

## 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] P. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.

[2] 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/.

[3] R. De By, W. Klas, J. Veijalainen (eds.), Transaction Management Support for Cooperative Applications. Kluwer Academic Publ., December 1997.

[4] 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.

[5] A. Elmagarmid, Database Transaction Models for Advanced Applications, Morgan Kaufmann, 1992.

[6] J. Gray, A. Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann, San Mateo, CA, 1993.

[7] 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.

[8] G. Letham, GIS Movers and Shakers Target LBS. http://spatialnews.geocomm.com/newsletter/2000/22/mobile2001l.

[9] Merita Solo. http://solo.merita.fi.

[10] Merita Solo over WAP services. http://www.merita.fi/s/kodin/osta/laskut/osoite_wap.stm.

[11] Mobile Electronic Transactions forum. http://www.mobiletransaction.org/.

[12] MeT Overview White Paper. Version 2.0, 29.1.2001. http://www.mobiletransaction.org/pdf/White%20Paper_2.0.pdf.

[13] Nordea bank. http://www.nordea.com/eng.

[14] C. Papadimitriou, The Theory of Concurrency Control, CS Press, Rockville, MD, 1986.

[15] J. Suutari, Transparencies on the Architecture of Location-based Services in WAP Environment, MMM-Nokia Meeting, November 2000.

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

[17] P. Tarasewitch, M. Warkentin, Issues in Wireless E-Commerce, ACM SIGEcom Exchanges, Issue 1.1, August 2000, pp. 19-25.

[18] P. Timmers, Business Models for Electronic Markets. http://www.electronicmarkets.org/netacademy/publications.nsf/all_pk/949/$file/v8n2_timmers.pdf?OpenElement&id=949.

[19] 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.

[20] J. D. Tygar, Atomicity in Electronic Commerce. In Proceedings of the 15th PODC Conference, 1996: 8-26.

[21] J. Veijalainen, Transaction Concepts in Autonomous Database Environments. (Ph.D. thesis). GMD-Bericht Nr. 183, R. Oldenbourg Verlag, Munich, Germany, April 1990.

[22] J. Veijalainen, A. Tsalgatidou, Electronic Commerce Transactions in Mobile Computing Environment. Q. Jin, J. Li, N. Zhang, J. Cheng, C. Yu, S. Nogushi (eds), Proc. of IS2000, Fukushima, Japan, Nov. 5-8, 2000, pp. 37-45.

[23] J. Veijalainen, F. Eliassen, B. Holtkamp: The S-transaction Model. Chapter 12 in [5]

[24] 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 Publ., San Francisco, October 1998, pp. 301-350.

[25] J. Veijalainen, Transactions in Mobile Electronic Commerce. In: G. Saake, K. Schwarz, C. Türker (eds.), Transactions and Database Dynamics. LNCSNr. 1773, Springer Verlag, Berlin, December 1999: 208- 229.

[26] E. Wesel, Wireless Multimedia Communications, Networking Video, Voice and Data. Addison-Wesley, 1998.

[27] Wireless Application Protocol. See WAP-forum at www.wapforum.org.

[28] Keltainen Pörssi at www.keltainenporssi.fi.

[29] Liquidaudio at http://www.liquidaudio.com/.

[30] J. Tang, and J. Veijalainen, On E-Commerce Transaction Protocols that support Atomicity Based Dispute Handling with Untrustworthy Players. In Proceedings of ICTEC 2000, Dallas, TX, USA, Nov. 16-19, 2000, pp. 299-314.

[31] 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.

[32] A. Devine, S. Holmqvist, Mobile Internet Content Providers and their Business Models. Masters Thesis, Stockholm Kungl Tekniska Högskolan, January 2001. http://www.japaninc.net/online/sc/master_thesis_as1.pdf.

[33] 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

[34] 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.

[35] WAP forum; WAP Specification 2.0. Accessible at http://www.wapforum.org/.

[36] Open Mobile Alliance. http://www.openmobilealliance.org/.

[37] 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.

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

[39] H. Garcia-Molina, K. Salem: Sagas. Proc. of ACM SIGMOD conference, May 1987, pp. 249-259.

[40] 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.

[41] 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.

[42] 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.

[43] Jari Veijalainen, Mathias 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. (forthcoming).