Open Philosophies for Associative Autopoietic Digital Ecosystems

Contract n° 034824

Workpackage 3: Autopoietic P2P networks

Deliverable D3.3:

Full Architecture Definition

Project funded by the European Community under the “Information Society Technology” Programme

Contract Number: 034824

Project Acronym: OPAALS

Title: Open Philosophies for Associative Autopoietic Digital Ecosystems

Deliverable N°: D3.3

Due Date: 05/2008

Delivery Date: 07/2008

Short Description:

This document is the third in the series of deliverables from WP3, which is focused on the development of a fully distributed P2P architecture to support open and trusted collaborations between SMEs to ensure their sustainability within a pan-European Digital Ecosystem.

The first deliverable detailed the OPAALS distributed transaction model and identified the demands it makes on the supporting P2P network, and set out the preliminary architecture required. This second deliverable focused on the formal analysis of both the transaction model and the characteristics of the underlying P2P network architecture.

This deliverable documents the overall status if the OPAALS work on the autopoietic P2P network. It brings together a complete description of the P2P architecture, including the transaction model that it central to the birth and growth models for the P2P network. In addition it includes reports on the extension of this work to Mobile Networks, and on a series of simulation experiments.

Authors: Surrey, SUAS, CreateNet, TechIdeas

Partners contributed:

Made available to: Public

VERSIONING		
VERSION	DATE	AUTHOR, ORGANISATION
1.0	**03/07/08**	**SURREY: P. KRAUSE; A. MARINOS; S. MOSCHOYIANNIS; A. RAZAVI; Y. ZHENG** **SUAS: T. KURTZ;** **CREATENET:** **TECHIDEAS: MIKALA P. STREETER & JESÚS E. GABALDÓN**
2.0	**05/08/08**	**SURREY: P. KRAUSE**

Quality check:

1st Internal Reviewer: Paul Malone (WIT)

2nd Internal Reviewer: Ossi Nykänen (TUT)

Table of Contents

1. Introduction

This report provides an overview of the current status of the OPAALS Architecture for an "Autopoietic Peer to Peer (P2P) Network". A key feature of the architecture is the generation of local collaboration networks. Peers from within these local collaborations then emerge that can be designated (for so long as their availability and capability allows) to act as points of communication with other local collaboration networks. As a result, a dynamic virtual super peer architecture emerges, but with a key feature that the role of super peer can and will be migrated to another location as soon as any physical node fails to offer the resources and performance required to meet its objectives. Super peers *per se* are not a new concept, and concerns have been raised about their use: according to [1], for example, "super-peer networks can alleviate the performance issues of pure unstructured topologies, but at the price of introducing two distinct peer classes". But the issues only arise if these are *static* super peers, with identifiable locations (aka "targets for attack"). The novelty of the OPAALS work with this respect lies in the very core of having *virtual* super-peers, demonstrating the ability to *dynamically* assign the "two distinct peer classes". Furthermore, the dynamic virtual super peers (DVSPs) consist of clusters of nodes taken from different organizations.

In that sense, the P2P architecture *appears* to be analogous to a democratic social system in which a community elects certain key representatives to act as points of communication to other communities of interest (and maintaining that role only so long as they maintain an appropriate level of trust and performance). However, this document only reports this work from the computer science perspective, and the links to studies of emergent properties in social networks are left for later work.

Section 2 provides an overview of the work that has been performed in WP3 in Phase I of OPAALS. We then proceed (Section 3) to discuss the generation of a specific form of local collaboration - a long-running e-commerce transaction in which a number of small enterprises collaborate to deliver a response to a consumer request. How to perform the responsive generation of a composite service that meets such a request is a key research question. Section 4 reports on work that is in hand to support the generation of transactions in response to declarative statements of what is required by a consumer.

Section 5 then provides an overview of the agent model that enables a digital ecosystem to emerge from the local collaborations needed to support service requests. This is followed (Section 6) by a report by work from SUAS on a series of simulations to analyse the properties of the transaction model.

All the proceeding work is abstracted from the specific issues that may arise in cases where a significant proportion of the emerging DE is situated on mobile devices. Section 7, from CreateNet, focuses down on the specific issues of DEs and Mobile Networks.
The final work presented, in Section 8, covers supporting experiments from TechIdeas on the creation of scale-free networks and simulation environments. We then conclude with some pointers to next steps.

[1] Nejdl, W., & Siberski, W, (2006). Schema-Based Peer-to-Peer Systems. In Steinmetz, R., & Wehrle,K. (Eds): Peer-to-Peer Systems and Applications. Lecture Notes on Computer Science, Springer, USA.

2. Overview of WP3 in Phase 1

The work in WP3 of OPAALS has focused on facilitating cooperation between SMEs and improving connectivity in a pan-European Digital Ecosystem.

Cooperation between SMEs can be modelled as long-term business activities involving a number of participants. These business activities can have overlapping actions (parts), which provide an opening for strengthening cooperation between participating SMEs in the long term. What is extremely important in this context is the preservation of local autonomy of each SME during a business activity so as to foster an environment of open and fair competition whose infrastructure does not allow large enterprises to gain unfair advantage and dominate the business setting.

To provide this open collaborative environment we have introduced local agents for SMEs in order to equip them with a cooperative business model that comes with full consistency on one hand, and durability on the other. At the same time, this design allows for optimal (or maximum) concurrency and recoverability. The structure of the coordinator in the local agent relies on an extended lock system for providing independent consistency, without violating local autonomy, log-based structures for ensuring global consistency, and a vector model for handling the complex interactions required in long-term business activities. The log system facilitates business activities in general (IDG) and handles overlapping activities in a consistent manner (EDG, for addressing partial results). Formal reasoning within the (vector) model of interactions is used to guide the compensating actions taken (local left-closure) and identify alternative scenarios (discreteness), and that is in addition to detecting missing acknowledgements which make deriving a UML model of the cooperative environment feasible.

One of the advantages of this design is that it addresses omitted results (preserving as much progress-to-date as possible) not only through the provision for forward recovery but also by providing fully isolated recoverability that forbids any propagation of inconsistent results. These consistent, durable, concurrent, recoverable long-term business activities conceptually are called transactions. When a transaction takes place it creates a temporary network of participating SMEs, which we have called Virtual Private Transaction Network (VPTN).

By their very nature these collaborative networks (VPTNs) provide diversity and dynamicity, which we used in creating a dynamic topology for a connected pan-European Digital Ecosystem. The VPTNs are connected to form the so-called Virtual Service Network (VSN) based on a measurement of reliability (stability function) that is calculated dynamically and champions a node from each VPTN. After particular consideration, and to avoid the issues regarding static super-peers flagged in p.45 of the DoW, we have opted for clusters of stable nodes, whose members can change at any time. The members of these Virtual Super Peers (VSPs) are determined continuously by the stability measurement which is based on dynamic properties of node availability. This allows any node to join a VSP while it also avoids violation of local autonomy and stirs the topology away from any centralisation point - these are characteristics that cannot be achieved using static super-peers protocols.

The VSPs can change at any time, and thus the virtual clustering of the network they manage also changes dynamically, which means that the network topology continuously evolves and adapts to the traffic, availability and other dynamic aspects of Digital Ecosystems such as the situation of nodes continuously joining and leaving the network. In order to ensure that the topology continuously evolves in a way that reinforces its good features and acts against those that act as threats, we have taken inspiration from biological studies of growth in molecular networks and in particular major evolutionary events such as domain duplication and edge innovation, which we apply at the network level.

Figure 2.1, below, provides a high-level overview of the OPAALS architecture/service stack. This report primarily focuses on the transaction support, and P2P network layers. In addition, we provide some

introductory material that is in hand to map service requests onto service compositions that deliver responses to the consumer request. We aim to address two research questions in this work:

1. Can we produce a model that is applicable to services and information resources by working within a "RESTfull" framework, where resources (nouns) are identified by URIs, and accessed by a uniform interface of well-defined methods (verbs): GET; PUT; POST; DELETE?
2. If we then combine this with "proper" declarative use of SBVR, can we generate responses to requests ("service compositions") in a general way without the need for predefined workflows?

The goal here is to free business from the inhibitions and constraints of pre-defined workflows, and open up the scope for innovative ad-hoc responses from an open pool of resources.

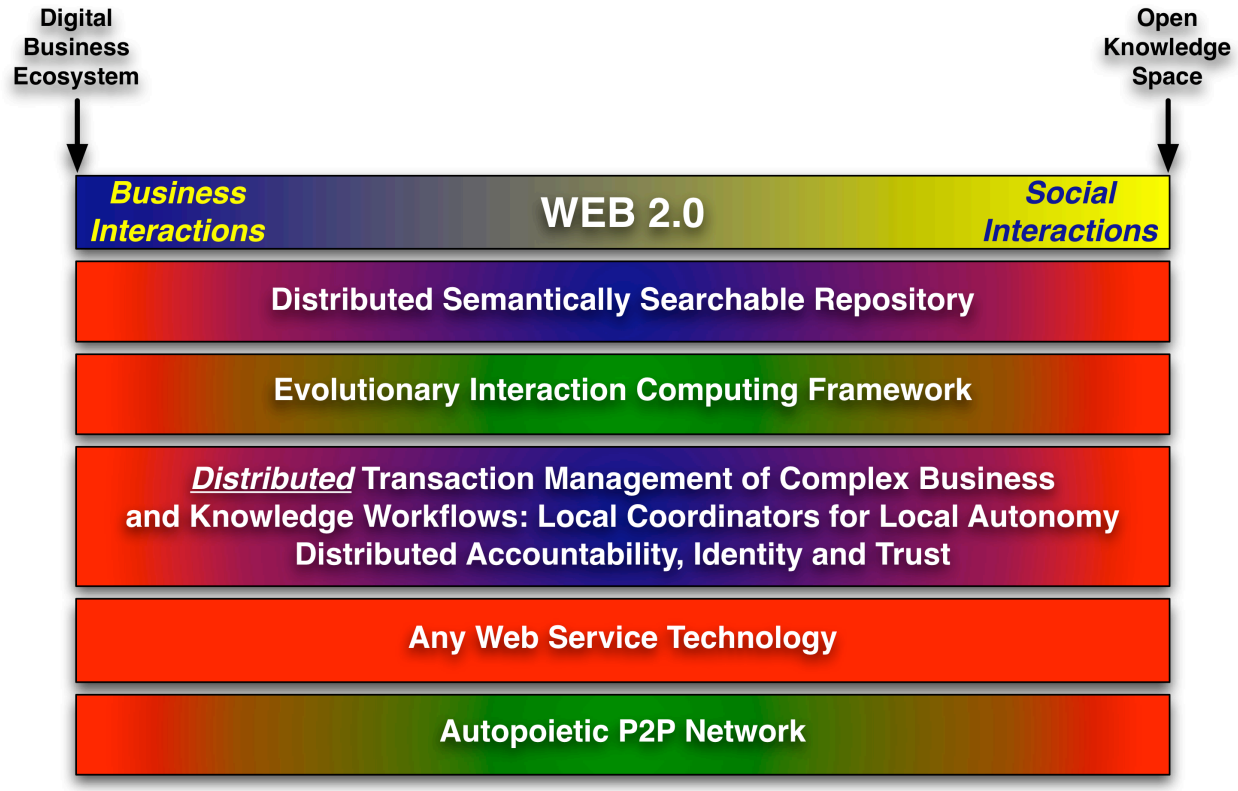


Figure 2.1: The OPAALS Service Stack

In order to help understand the motivation behind many of the design choices in the sequel, we finish this part with some hints from Description of Work:

What should the Autopoietic P2P network provide?

"*the Autopoietic P2P network provides a self-maintaining environment that will configure itself to support the survival of those concepts or entities that reside within it, and who contribute to its purpose and continued survival, and acts to repel those entities which act as threats.*" **DoW Page 45, first paragraph line 4th**

The main requirements of such a network are?

"*The main requirements for such an architecture come from the transaction models that must support long-lived transactions mediated by automatically and dynamically composed service chains, and that will be examined in the second half of Phase I. The human and computational dimensions of identity, accountability, and trust are addressed in the next workpackage.*" **DoW Page 83 last paragraph, line 7th**

More details about P2P architecture:

"*Another thread of activity will begin with the study of scale-free networks of variable topology and the development of a truly distributed P2P architecture. Some of this work has already been started in the DBE integrated project and has provided an initial run-time architecture for the DBE Execution and Evolutionary Environments. In OPAALS the dynamic nature of such networks will be developed further by endowing them with algorithms that support self-organising and autopoietic behaviour. Some of this work will be mathematical and some will be closer to computer science. An important output of this WP will be a P2P architecture that is compatible with a transactional model that can support long-lived transactions between automatically composed chains of services.*" **DoW Page 82 Second paragraph**

Dynamic topology of the scale-free P2P network:

"*The aspect of digital ecosystems that most closely evokes the architecture and function of the brain is the dynamic topology of the scale-free P2P network. It is the sixth objective of the project to investigate the most effective P2P architecture that can support adaptive and learning behaviour of the digital ecosystem. In particular, this work aims to provide a dynamic P2P network architecture and topology (WP3) that can support seamlessly and efficiently the transaction models….*" **DoW Page 6 third paragraph, line 7th to the end.**

And this has been re-mentioned:

"*…It is the sixth objective of the project to investigate the most effective peer-to-peer architecture that can support adaptive and learning behaviour of the digital ecosystem. In particular, the work in WP3 aims to provide a dynamic network architecture and topology that can support seamlessly and efficiently the transaction models …*" **DoW Page 45, second paragraph line 5th**

Avoiding static topology, using of Super peers (nodes) and/or any sort of centralised point:

"*...key requirements for a P2P network that supports digital ecosystems are that the network should demonstrate no critical dependencies on single organisations, and have no critical points of failure. This means that we must move away from super-peer networks in which the super-peers are static and physically reside on servers (or clusters of servers) owned by a single organisation. The environment in which the ecosystem resides must be truly distributed and dynamic.*" **DoW Page 45, second paragraph line 7th**

Concept of autonomous and self-maintaining unity:

"*The P2P network thus becomes an "autonomous and self-maintaining unity" [Principia Scientifica Web, 2005] that provides an environment that will guarantee not only its own survival, but also that of the concepts or entities which reside within it.*" **DoW Page 45, first paragraph**

Avoiding any boundary and Necessity of using Virtual Super Peers:

"*We must not impose any boundaries on the growth of the network; hence, of course, it must be scalefree. That requires us to maintain the concept of super-peer. But rather than pre-assigning the ownership and location of the super-peers, we now introduce the notion of a "**virtual super-peer**" whose creation is spontaneously generated as and when needed, and whose physical location may change dynamically depending on the network's needs...*" **DoW Page 45 last paragraph**

The Techideas role in WP5:

"*The outputs of WP2, 3, and 4 will be tested and implemented in the evolving digital ecosystem infrastructure by Techideas (WP5).*" **DoW Page 82 third paragraph**

Is it possible to have an alternative view?

"*An alternative view of a digital ecosystem is as a heterogeneous set of "digital species" that are competing against each other subject to a selection pressure that comes from the users. The combination of many such individual optimisation processes can be argued to lead to an overall adaptation and evolution*

of the ecosystem to the needs of its users, improving the utility that each SME derives from the digital ecosystem." **DoW Page 6 last paragraph.**

2.1- Task 3.1 – Propagation models for scale-free networks (UniS)

Within the context of Task 3.1, the transaction model developed in OPAALS (as promised in p.83 of the DoW) is based on the notion of local coordinators who perform distributed coordination of the interactions involved in a transaction whilst respecting the loose-coupling of the underlying services and the local autonomy of the participating platforms. This (local coordination) allows the bulk of the messages to be propagated along the Participants without going through the Initiator of a transaction in each and every case. The distributed coordination is reflected in the resulting Virtual Private Transaction Networks (VPTNs), which in turn gives rise to the Virtual Service Network (VSN) (connected VPTNs based on stability function). Because the amount of traffic of each coordinator is variable, the traffic can change the outcome of the stability function (which provides essentially a measurement of the overall network reliability). Consequently, the role of the node within its corresponding VPTN will change, e.g. when a node experiences heavy traffic overload, the stability measurement decreases (more interactions come with larger probability of disconnections/failures) and thus the node can no longer be considered for a virtual super peer (VSP) – it will be replaced by a more stable node (with less traffic) from that VPTN in the corresponding VSP. This dynamicity, reflected in the propagation model of the VPTN, shapes the dynamic topology of the P2P network through the notion of VSPs and virtual clustering described in detail in D3.2.

2.2- Task 3.2 – Dynamic mapping from Virtual to Physical Topology, and Transactional Models (UniS)

A key requirement for the P2P network (as reported in the DoW) is that the physical location of the super peers must be able to change dynamically, depending on the reliability, availability and loading of the servers in the network. It turns out that the dynamic topology of the Dynamic Virtual Super Peers (DVSPs) discussed under Task 3.1 above, automatically accounts for the mapping between virtual and physical topology. Indeed, a key feature is that the physical location of the DVSPs is not discoverable by any participant in, or user of, the P2P network. The relationship between the Virtual Private Transaction Networks and the dynamic identification of DVSPs for maintaining the network topology are discussed in Section 5 of this report.

2.3- Task 3.3 – Mathematical Models of Autopoietic P2P networks (UniS)

With respect to Task 3.3 we have developed a formal model for the coordination of distributed transactions (VPTNs), drawing upon the mathematics of abstract algebra, order theory, set theory and logic. The formal model allows for a thorough understanding of the behaviour patterns the underlying service compositions should follow in order to guarantee a successful outcome – a transaction either commits or is compensated for. Also, it can reveal the alternative scenarios necessary for including provision for forward recovery. The mathematical model developed is based on an algebra of vectors (in a tuples-based representation of behaviour) that exhibits true-concurrency and has been shown to have important compositional properties. The latter are exploited in reasoning about global behaviour based on the behavioural analysis of the local coordinators. In this way, the formal analysis of the interactions present in the autopoietic P2P network can be lifted from the VPTN level to the VSN level, via the (dynamically formed) virtual super peers (VSPs). This formal analysis has lead to the identification of the actual break-points for creating VSPs – given by the measurement of stability, as a function of actual, versus promised, availability and overall reliability over time.

2.4- Task 3.4 – The Evolutionary Framework (SUAS)

Adaptation and preparation of the EvESim for service calls and nested service calls in serial and parallel was achieved. Thus, service calls can be simulated based on a XML based service transaction description.

Successful implementation of the service calls in a distributed simulation took place, including parsing of transaction trees coming from UniS. This point was only partially implemented in order to be able to test a distributed service execution with the UniS transaction model. More work has to be done to fully support their model – partial results and rollback scenarios are not currently possible, for example.

3. Long-running Transactions: semantics, schemas, implementation (UniS: S. Moschoyiannis, A. Razavi, Y. Zheng, P. Krause)

In this section we describe work that has been going on with respect to developing a distributed model for coordinating long-running transactions and handling possible failures during the course of the execution of the underlying service compositions. As discussed in Section 2, the main requirements for the P2P network architecture sought after in WP3 of OPAALS come from the need to support long-lived transactions between networked organisations, and that is without requiring a single point of control or being amenable to a single point of failure. The WP3 tasks performed in Phase I of OPAALS were aimed at investigating the most effective architecture for supporting seamlessly and efficiently the distributed transaction model, and will be described in detail in the remaining sections of this report.

The need for providing robust support for open e-business transactions between SMEs arose in the late stages of the Digital Business Ecosystem (DBE) project [19]. Preliminary ideas on how transactions between participating SMEs can take place in distributed manner appeared in deliverable D24.5 of DBE [8]. Work in this direction continued in OPAALS, specifically aiming to support long-lived business activities that correspond to long-running transactions whose execution involves a number of underlying interaction-based service compositions which need to be performed in a principled manner, and in a way that their effects can be 'undone' in case some subsequent failure (due to the current unavailability of a service or resource, for example) in the course of the transaction makes this necessary. All the while this should be achieved without violating the local autonomy of the participants or creating dependencies on a central point of control or a single point of failure.

In what follows we give an overview of the work done on the transaction model in Phase I, in terms of the key concepts that ensure the desired characteristics mentioned above, at the design level (transaction trees, log structures) and the formal modelling level (transaction vectors) whose combination provides a language in which to express the required distributed coordination of the underlying services. In addition, we provide schemas that capture the formal semantics underpinning the design that can be sued to guide the implementation. Much of this work has already been reported in D3.1 and D3.2 of OPAALS [23]. However, we provide this overview for completeness. In addition, it does contain updates, including the schema definitions and the example implementation. In that regard, it provides a description of the current evolution of the support for long-term transactions.

3.1- Long-running transactions in DEs

A business transaction between SMEs in a Digital Ecosystem can be either a simple usage of a web service or a mixture of different levels of composition of several services from various service providers. The specification of a business transaction may allow it to be completed over a period of minutes, hours, or even days – hence, the term long-lived or long-running transaction. The execution of a long-running transaction corresponds to conducting a business activity and typically comprises a number of sub-transactions or activities that involve the execution of a number of underlying services.

This makes Service-Oriented Computing (SOC) [1], whose goal is to enable applications from different providers to be offered as services that can be used, composed and coordinated in a loosely-coupled manner, the prevalent computing paradigm in a digital business ecosystem. The actual architectural approach of SOC is called SOA [2] and is particularly applicable when multiple distributed applications are running on varied technologies and platforms need to communicate with each other.

Business transactions in a business-to-business (B2B) context typically involve interactions between multiple partners, either service providers, or service consumers, or both. This requires that all partners behave in a coordinated manner – partners must follow some protocol to execute a transaction effectively.

Conventional transaction models such as Sagas [3] or the more recent models targeting web services Web Services Transactions (WS-Tx) [4] and Business Transaction Protocol (BTP) [5] seem to be geared towards centralised control (based on the WS-Coordination framework [6]) which means some knowledge of the internal build-up of the participants is required. This violates the primary requirement of SOA for *loosely-coupled services* and may not be acceptable or even possible in a business environment – in a digital ecosystem involving communities of SMEs this raises a barrier for the adoption of SOA as it violates local autonomy. A comprehensive discussion of a range of existing transaction models can be found in Deliverable D24.5 of the DBE project (see [19]). A more thorough discussion on the shortcomings of WS-Tx and BTP has been included in Deliverable D3.1 and D3.2 of OPAALS [23].

Part of the problem seems to be that these frameworks lack a formal model for the coordination of the underlying services involved in the execution of a long-running transaction. Further consequences of this may come to view when a transaction needs to be compensated due to a subsequent failure.

3.2- Transaction model for DEs

In deliverable D3.2 of OPAALS [23] we have described a distributed transaction model for networked organisations in the digital ecosystem paradigm. The objective is to provide robust support for distributed long-term transactions between SMEs, and this has been pursued at the modelling, design, formal (reasoning) and implementation (schemas, programming) levels. This work has been subsequently been published in [7], [20], [21], [22]. In the remainder of this chapter we only outline the basic ideas. The presentation draws upon the high-level overview of the proposed transaction model and the formalisation of the key concepts found in [20].

It is often the case that internal activities of a transaction need to share results before the termination of the transaction (transaction commit). More generally, dependencies may exist between activities inside a transaction due to the required ordering on service invocations (e.g. book a hotel only after booking the flight) or due to the sharing of data (one service execution uses the results of another). The use of log structures, given in the form of directed graphs, has been described in [7], [8] for capturing the internal and external dependencies of a transaction.

For example, if a transaction involves the sequential composition of services, this gives rise to a dependency (due to ordering and / or data) between the corresponding service invocations. As a consequence, if one is aborted for some reason, then the other also needs to be aborted in order to maintain consistency across the transaction.

In [17] we have introduced the so-called Internal Dependency Graph (IDG) for representing such dependencies. Similarly, an External Dependency Graph (EDG) is used for representing dependencies between services from different transactions in a digital ecosystem for business. Fig. 3-1 shows the IDG for pairs of services that have been composed using different types of composition, namely Sequential with Data Dependency (SDD) and Parallel with Data Dependency (PDD).

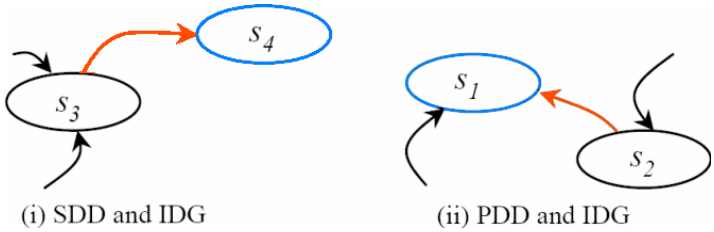


Figure 3-1 Internal Dependency Graph for sequential and parallel service composition

The benefit of using these lightweight logs is that each platform coordinator only needs to have knowledge about its services and their dependencies to other coordinators' services – it needs to know only what happens *before and after* its own services. This is to avoid forcing service providers to reveal more than they want to reveal about their services, processes, data, implementation designs, and ultimately their business models.

In our model a transaction is represented by a tree structure that allows for nested subtransactions (representing smaller, internal activities that need to be performed within the context of a larger business activity) and exemplifies the local coordination that is required for the services involved to be performed in unison.

Drawing upon the latest work on an extended service-oriented architecture for a business environment [2], we have considered five different types of coordinators which allow for various modes of service interaction in our model. In the transaction tree of Fig. 3-2, the services s3 and s4 for example are children of a sequential coordinator and hence, s4 can only be executed after s3. In other words, the execution of s4 is dependent on the (successful) execution of s3, e.g. s4 uses the results released by s3.

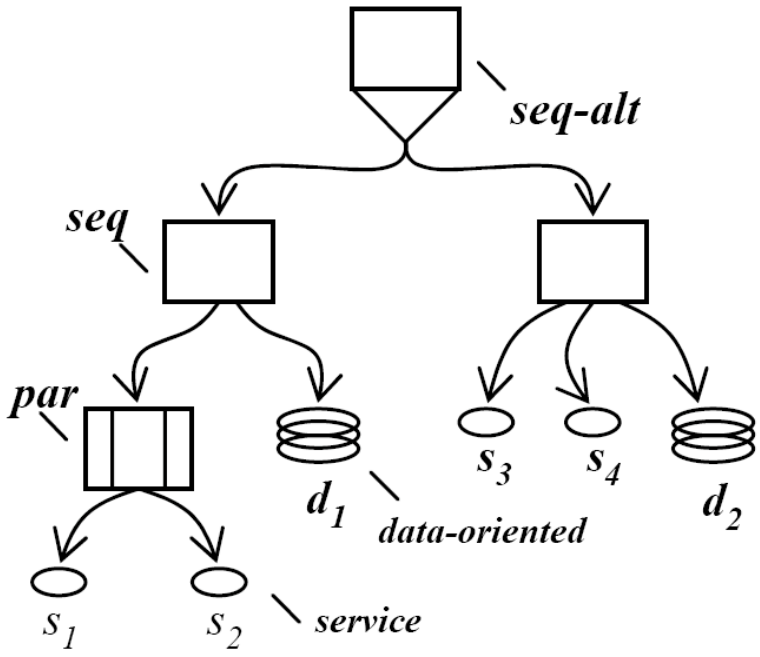


Figure 3-2 Transactions in a tree structure

The scenario described in Fig. 3-2 has appeared in [7] and has been simplified here somewhat. Nevertheless the transaction in question is complicated enough to illustrate the key ideas of our formal modelling approach to long-running transactions.

The DE nested transaction model can be represented in a context schema. This is used by the Initiator of a transaction to specify the different levels of compositions of the underlying services. The schema also shows the boundaries and requirements of the model and the need for a formal language to describe transactions, which will be outlined in Sections 3.2.1 – 3.2.4. Meanwhile, Fig. 3-3 shows such a schema (the Netbeans 5.5.1 source used to create the associated schemas can be found following the link of [9]).

The nested transaction structure is imposed in the schema using the 'Coordinator' element which can include a service composition ('Composition' element in the schema) or usage of a simple web service (notice the 'WebService' element in the schema), including a purely data-oriented web service (a leaf of the tree). Each service composition ('Composition') has a type ('CompositionType'), which determines the mode of the interaction-based composition, whether it is sequential, parallel or alternative, and we will see how each is treated formally in our approach in Section IV. It has two or more 'Coordinator' elements which introduce recursion - another service composition or a simple web service. This recursive definition can generate an XML representation of the transaction tree. In the following sections we interpret this context schema into a formal language which allows for an XML presentation of the semantics of a transaction which can then be implemented in different platforms.

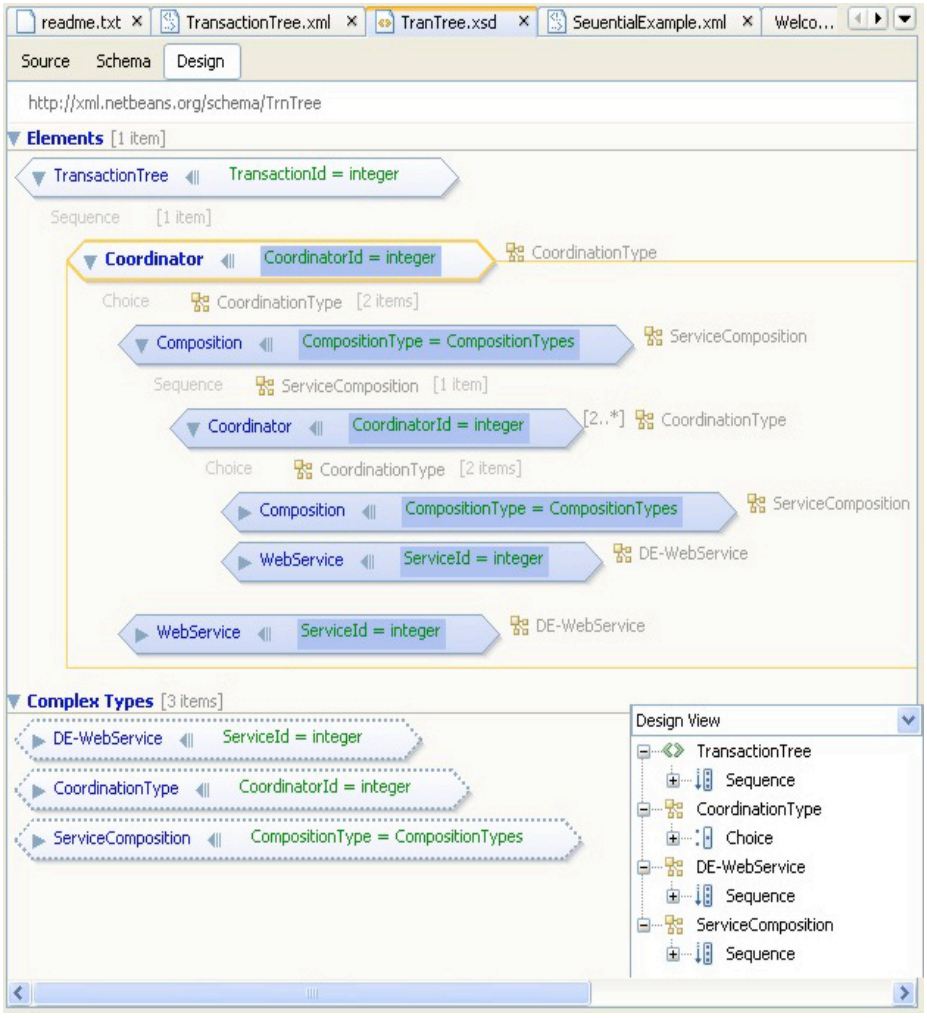


Fig. 3-3 Transaction Context Schema

3.2.1. Formal description for long-running transactions

In this section we introduce a formal language for describing long-running transactions. Apart from the dependencies between service executions, there is a high degree of concurrency in a transactional environment since real problems require a number of activities to take place in parallel. For this reason we will find the general theory of non-interleaving representation of parallel behaviour found in [10] of great use in what follows. As mentioned before, our objective is to get a thorough understanding of the behaviour the underlying service compositions need to exhibit for a successful outcome of the transaction as a whole.

The semantics is intended to describe the behaviour of a transaction in terms of its services at the deployment level, but not the low-level computations performed by the services themselves. Note that services in a digital ecosystem for business are offered from different service providers and it is important that we defer from interfering with the local state of the service execution. The adoption of a service-oriented architecture for distributed transactions reinforces our interest in all environmentally observable actions that take place during the course of the execution of a long-running transaction. This means it is appropriate to consider that any action within the transaction model has no significant duration, in the sense that (i) it either occurs as a whole or not at all; (ii) it occurs either wholly before, or wholly after, or wholly in parallel with, every other action.

A transaction may thus be associated with a finite set of events or significant events or actions that may occur during execution, e.g. service invocation, initialisation, commitment, service return, release result (return), termination, abort, etc. We denote this set of actions of a transaction by *M*.

A transaction *T* is associated with a set of leaves or access points *L* which consists of a set of basic services *S*, a set of data-oriented coordinators *D* and a set of delegation coordinators *Dlg*. Hence, $L = S \cup D \cup Dlg$. We further require that the sets *S*, *D* and *Dlg* are pairwise disjoint.

Actions within a long-running transaction take place on the leaves (of transaction trees) and therefore each leaf is in turn associated with a set of actions that may occur on that leaf, depending on its nature. We denote this set by $\mu(l)$, for each leaf $l \in L$, and require that $\bigcup_{l \in L} \mu(l) \subseteq M$, so is μ an *indexed cover* [11] for the set *M*.

As can be seen in Fig. 2 a transaction has a number of activation points, namely the leaves of the corresponding transaction tree. Thus, instead of modelling a transaction by a sequential process that would generate a trace of a single access point, we model the behaviour of a transaction by considering a set of such sequences at the same time, one sequence for each leaf. This draws upon Shields' *vector languages* [10] and allows us to capture what is happening on each access point of a long-running transaction.

Transaction vectors. Let *T* be a transaction. We define V_T to be the set of all functions $\underline{v}: L \rightarrow M^*$ such that $\underline{v}(l) \in \mu(l)^*$.

By $\mu(l)^*$ we denote the set of finite sequences over $\mu(l)$. Mathematically, the set V_T is the Cartesian product of the sets $\mu(l)^*$, for each *l*. Effectively, transaction vectors are *n*-tuples of sequences where each coordinate corresponds to a leaf in the transaction tree (hence, *n* is the number of leaves) and contains a finite sequence of actions that have occurred on that leaf or access point of the transaction.

When an action occurs on a leaf of the transaction tree, that is to say when an action associated with some subtransaction takes place on an access point, it appears on a new transaction vector and at the appropriate coordinate. For example, the vector ($s1$, Λ, Λ) describes that portion of behaviour of the transaction in which an action $s1$ (e.g. service invocation) has taken place on the corresponding service allocated to the first coordinate. We use Λ to denote the empty sequence.

The vector $(s1, s2, \Lambda)$ describes that portion of behaviour in which both *s1* and *s2* have happened on the corresponding services while the vector $(s1s3, s2, \Lambda)$ describes an occurrence of *s1* and an occurrence of *s3* on the service corresponding to the first coordinate, and an occurrence of *s2* on the second coordinate. Nothing has happened on the service corresponding to the third coordinate.

In can be seen that each transaction vector provides a snapshot of behaviour in which the transaction has executed the actions appearing on the vector's coordinates – it describes what actions have already occurred and on which part of the transaction tree. This vector-based description of behaviour allows to record the actions of a transaction as these occur on the multiple services involved in its execution. Readers familiar with process algebras like CSP or CCS can understand each particular coordinate of the vector description as a sequential CSP process. In this sense, the transaction vectors can be understood as the Cartesian product of sequential processes describing each leaf in a transaction tree.

Fig. 3-4 shows the xml schema types for building a single transaction vector. Each transaction vector comprises a specific number of 'access points' (five in our example), and each access point corresponds to a service, a data-oriented coordinator or a delegation coordinator. All are presented as 'DE-WebServices' in our xml schema.

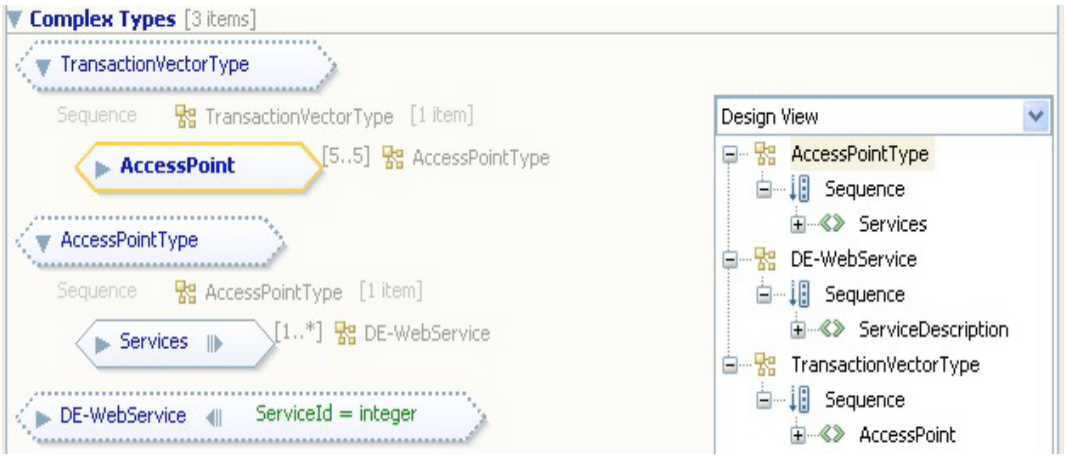


Fig. 3-4 A single vector type (xml schema presentation)

It can be seen from the example given above that there is already an ordering among actions on a particular access point or subtransaction, e.g. *s1* followed by *s3*. This vector-based behavioural description of transactions can also capture the orderings between different subtransactions, which amounts to actions appearing on different vector coordinates. This requires however a more careful consideration of the mathematical properties of such vectors which we briefly describe in the sequel.

Before examining the mathematical properties of our construction so far, we introduce a specific kind of transaction vector, which is used in our model to describe actions (events or activations) within a transaction.

Column vectors. Let T be a transaction and V_T its set of transaction vectors. We define

$$A_T = \{\underline{a} \in V_T \setminus \{\underline{\Lambda}_T\} : l \in L \Rightarrow |\, \underline{a}(l) \,| \leq 1\}$$

where $|\, x \,|$ denotes the length of sequence x. We refer to elements of A_T as *column vectors*.

Thus, column vectors are themselves transaction vectors, but have the additional constraint that each of their coordinates is either the empty sequence or a single action. For example, the vector $(s1, \Lambda, \Lambda)$ represents the occurrence of an action *s1* on the service associated with the first coordinate.

We will use the term transaction language to refer to a subset V of all possible vectors V_T formed over a given transaction T. Hence, a transaction T comes with a *language* V, where $V \subseteq V_T$. The idea is that the particular set of transaction vectors for a specific transaction expresses the ordering constraints necessary in the corresponding service orchestration.

We have seen that transaction vectors are essentially tuples of sequences. This can be exploited in defining operations on vectors in terms of well-known operations on sequences.

Let us establish some notation. If x and z are sequences, we write $x.z$ for the concatenation of x and z. As is well known this operation on sequences is associative with identity Λ, where Λ denotes the empty sequence. We also have a partial order on sequences given by $x \leq z$ if and only if there exists a sequence y such that $x.y = z$, and this partial order has a bottom element Λ. It is also well-known that the operation '.' is cancellative, which means that if $x \leq z$, then the sequence y such that $x.y = z$ is unique. We shall denote this sequence by z / x. Finally, recall that if x, y, z are sequences such that $x, y \leq z$, then either $x \leq y$ or $y \leq x$.

We may now lift these well-known operations on sequences onto transaction vectors. This is done formally in the following definition.

Operations on vectors. Let $\underline{u}, \underline{v} \in V_T$ be transaction vectors, we define

- $\underline{u}.\underline{v}$ to be the unique vector $\underline{w}$ such that $\underline{w}(l) = \underline{u}(l).\underline{v}(l)$, for each $l \in L$ (*concatenation*)
- $\underline{u} \leq \underline{v}$ iff $\underline{u}(l) \leq \underline{v}(l)$, for each $l \in L$ (*prefix ordering*)
- $glb(\underline{u},\underline{v})$ to be the vector $\underline{w}$ such that $\underline{w}(l)=\min(\underline{u}(l),\underline{v}(l))$, for each $l \in L$
- $lub(\underline{u},\underline{v})$ (if it exists) to be the vector $\underline{w}$ such that $\underline{w}(l)=\max(\underline{u}(l),\underline{v}(l))$, for each $l \in L$
- if $\underline{u} \leq \underline{v}$, then we define $\underline{v} / \underline{u}$ to be the unique element $\underline{z} \in V_T$ such that $\underline{u}.\underline{z} = \underline{v}$ (*right-cancellation*)

Thus, the operation of concatenation on vectors is defined in terms of the concatenation of sequences appearing on their respective coordinates. For example,

$$(s_1 s_3, s_2, \Lambda).(\Lambda, s_4, \Lambda) = (s_1 s_3, s_2 s_4, \Lambda)$$

The ordering amongst vectors is defined in terms of the usual prefix ordering operation on sequences appearing on their coordinates. For example, $(s1, s2, \Lambda) \leq (s1s3, s2, \Lambda)$ since $s1 \leq s1s3$ and $s2 \leq s2$ and $\Lambda \leq \Lambda$. In other words, the second vector 'wins' on the first coordinate (since it has a sequence of greater length in this coordinate) while the two vectors draw on all other coordinates.

It is not hard to see that some vectors will be incomparable. It turns out that such vectors describe either parallel or alternative behaviours of the transaction in question, and this will be further discussed in the following sections.

The operations *glb()* and *lub()* give the greatest lower bound and the least upper bound, respectively of $\underline{u}, \underline{v} \in V_T$, in the usual sense of lattices and domain theory [11]. As we will see, these operations are central to the treatment of concurrency in our approach.

The right-cancellation operator '/' says that if $\underline{u}$ is a transaction vector describing an initial part of the behaviour described by $\underline{v}$ so that $\underline{u} \leq \underline{v}$, then $\underline{v} / \underline{u}$ is the 'continuation' of $\underline{u}$ that extends it to $\underline{v}$. This operation is central to the treatment of compensations in our approach. We do not deal with compensations as such in this section, but will have more to say about expressing compensating behaviour in the sequel

It is important to stress the fact that all operations on vectors are performed coordinate-wise and this simplifies the proofs but also makes the formal model feasible for implementation.

Transaction vectors can be seen to be built up from the empty vector $\underline{\Lambda}_T$ by a series of concatenations with column vectors that represent actions. In fact, in describing the behaviour of a transaction we are interested only in those vectors describing (orderings of) actions that we expect the transaction to engage in during the course of its execution. This is the subset of all possible transaction vectors, over a given T, we referred to as the *transaction language*.

3.2.2. Sequential service composition

The prefix ordering relation on transaction vectors can be viewed as an ordering on partial executions, where each vector corresponds to that portion of behaviour in which the transaction has already engaged in the actions appearing on its coordinates. This can be expressed more succinctly by saying that $\underline{u} \leq \underline{v}$ in a transaction language means that $\underline{u}$ is an earlier part of behaviour leading to $\underline{v}$.

A more careful examination of the mathematical construction shows that we can say more than that. Indeed, we find it useful to determine immediate predecessors (or successors) of a transaction vector.

Cover. Suppose that $\underline{u}, \underline{v} \in V \subseteq V_T$. We say that $\underline{v}$ *covers* $\underline{u}$ in V, and we write $\underline{u} \lhd_V \underline{v}$, if

(i) $\underline{u} \leq \underline{v}$ and $\underline{u} \neq \underline{v}$ and

(ii) If $\underline{z} \in V$ such that $\underline{u} \leq \underline{z} \leq \underline{v}$, then $\underline{z} = \underline{u} \vee \underline{z} = \underline{v}$.

Thus, whenever $\underline{u} \leq \underline{v}$, and we also have that $\underline{u} \lhd \underline{v}$, then the last actions that went into forming each vector have occurred in sequence - one after the other. This allows to model sequential dependency among services inside a transaction. Recall the example of Fig. 2 where service *s3* feeds service *s4*.

Fig. 3-5 presents the corresponding schema for sequential service composition. The 'TransactionVectors-Type' has a TransactionVector element which includes a single transaction vector (e.g. describing the service invocation *s3*) and the immediate successor (describing the service invocation *s4*). The latter is captured by the 'FollowedBy' element. The type of 'SequentialComposition' simply includes a single transaction vector followed by another transaction vector.

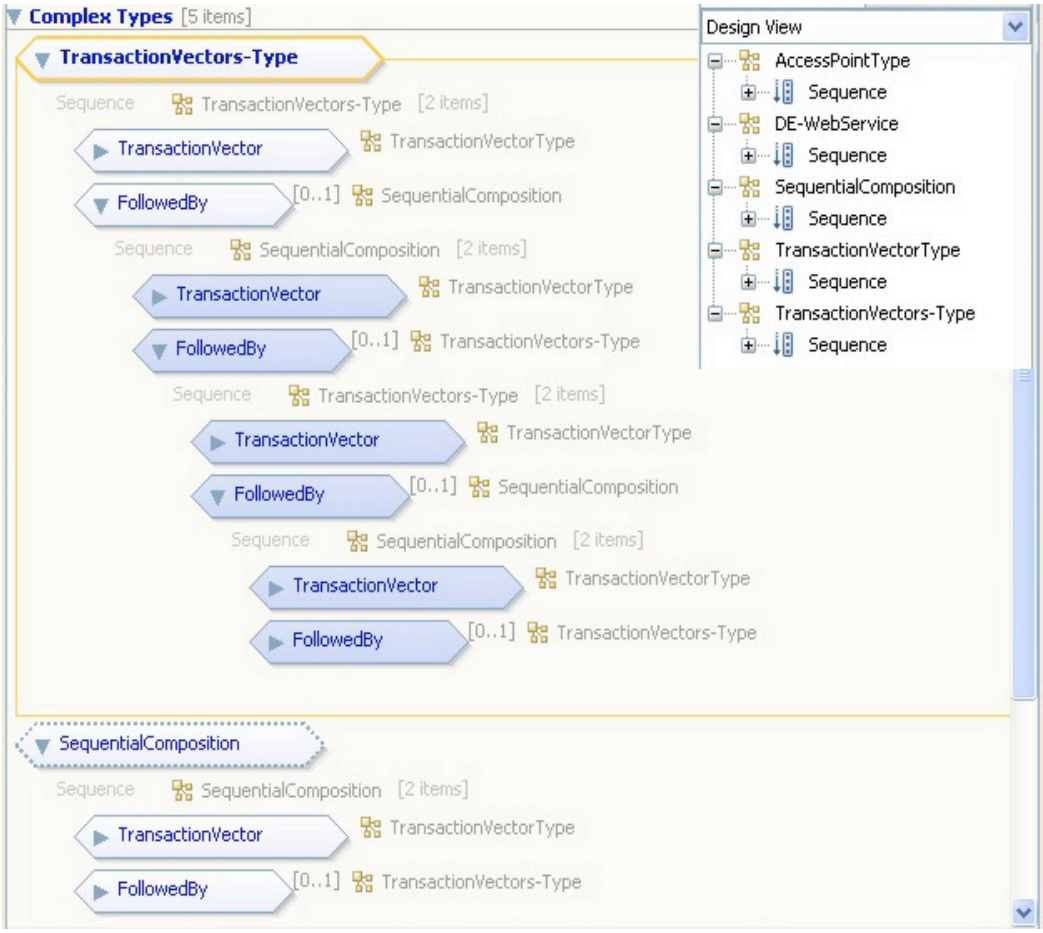


Fig. 3-5 Sequential service composition (xml schema presentation)

3.2.3. Parallel service composition

Our approach towards modelling concurrent actions, actions that can happen in parallel, draws upon the concepts in Shields' vector languages [10] and Mazurkiewicz trace languages [12] where concurrent events are considered as being unordered, in contrast to CSP trace theory where it is assumed that observations are sequential in nature and concurrent events are understood to occur in either order (nondeterministic interleaving).

The treatment of concurrency within our formal model of transactions thus takes up on non-interleaving models of concurrency, which introduce additional structure into formal languages in order to describe non-sequential behaviour. The additional structure is given in terms of an independence relation over action symbols, which describes potential concurrency.

In fact, the independence relation is a binary relation, defined over a set of actions, that is *symmetric* (to reflect the fact concurrency is always mutual) and *irreflexive* (to prohibit considering an action as being concurrent with itself). Intuitively, the independence relation ι defined over a set of actions A gives rise to an equivalence relation on sequences formed over A. Now we have seen that transaction vectors are essentially tuples of sequences, hence we may make use of this construction in terms of the sequences formed over the sets of actions $\mu(l)$, for each $l \in L$, of a transaction.

In terms of our notation it is appropriate to say that the independence relation on the set of actions A of a transaction equates all, and only those, sequences over $\mu(l)$, for each $l \in L$, which differ in the order of

adjacent and independent actions. Note that when the independence relation is empty in the sets $\mu(l)$, for each $l \in L$, no actions can be concurrent in the corresponding sequences $\mu(l)^*$, for each $l \in L$, which amounts to our understanding of sequential transaction processing systems (e.g. as described by processes in CSP and its extension with compensations in [15]).

Drawing upon the extension of the independence relation ι to *behaviour vectors* in [10], the notion of independence between actions in Mazurkiewicz traces can be readily interpreted into transaction vectors in our approach.

Independence. For $\underline{u}, \underline{v} \in V \subseteq V_T$, we define

$$\underline{u} \; ind \; \underline{v} \quad \Leftrightarrow \forall l \in L : \; \underline{u}(l) > \Lambda \Rightarrow \underline{v}(l) = \Lambda$$

This definition says that two transaction vectors are independent if the behaviours they describe concern distinct services (correspond to activation on different leaves of the corresponding transaction tree). This means that the behaviours described by u and v may occur independently.

In the case of column vectors, independence captures the fact that actions appearing in one vector may occur independently of those appearing in the other. If in addition the vectors representing these actions are adjacent in an expression (of the series of concatenations that went into forming the corresponding transaction vectors), then the actions are concurrent. Thus, whenever two actions are independent and are both enabled (can both occur at some point, after some behaviour) then, their corresponding column vectors commute, i.e. a1.a2 = a2.a1, and in the resulting behaviour the two actions are concurrent.

For example, suppose that a transaction with 3 leaves or 3 different access points for the services involved, has experienced a fragment of behaviour described by $\underline{u} = (\Lambda, \Lambda, \Lambda)$ and after that may engage in $\underline{a}_1$ and $\underline{a}_2$ concurrently, where $\underline{a}_1 = (s1, \Lambda, \Lambda)$ represents an invocation of service *s1* and similarly $\underline{a}_2 = (\Lambda, s2, \Lambda)$ represents an invocation of service *s2*.

We make the observation that $\underline{a}_1$ ind $\underline{a}_2$ (by definition of the independence relation given earlier) and consequently,

$\underline{a}_1 . \underline{a}_2 = (s1, \Lambda, \Lambda).(\Lambda, s2, \Lambda) = (s1, s2, \Lambda) = (\Lambda, s2, \Lambda).(s1, \Lambda, \Lambda) = \underline{a}_2 . \underline{a}_1$

Thus, we have $\underline{u} . \underline{a}_1 . \underline{a}_2 = \underline{w} = \underline{u} . \underline{a}_2 . \underline{a}_1$.

Indeed,

$\underline{u} . \underline{a}_1 = (\Lambda, \Lambda, \Lambda).(s1, \Lambda, \Lambda) = (s1, \Lambda, \Lambda) = \underline{v}_1$

and

$\underline{v}_1 . \underline{a}_2 = (s1, \Lambda, \Lambda).(\Lambda, s2, \Lambda) = (s1, s2, \Lambda) = \underline{w}$

We also have that

$\underline{u} . \underline{a}_2 = (\Lambda, \Lambda, \Lambda).(\Lambda, s2, \Lambda) = (\Lambda, s2, \Lambda) = \underline{v}_2$

and

$\underline{v}_2 . \underline{a}_1 = (\Lambda, s2, \Lambda).(s1, \Lambda, \Lambda) = (s1, s2, \Lambda) = \underline{w}$

In the resulting behaviour w the actions *s1* and *s2* are concurrent. The situation is depicted in the familiar diamond (or lozenge) appearing in Fig. 3-7(iii).

The parallel service composition schema is depicted in Fig. 3-6. A 'ParallelComposition' consists of two or more (up to the number of its access points, in this example 5) transaction vectors, followed by a 'TransactionVectors-Type', which imposes that all transaction vectors describing parallel executions have one immediate successor (their *lub()* identified in the schema by the 'FollowedBy' element) and the concurrent service invocations can happen on different access points (recall the independence relation - two actions are independent, and potentially concurrent, if they engage different access points).

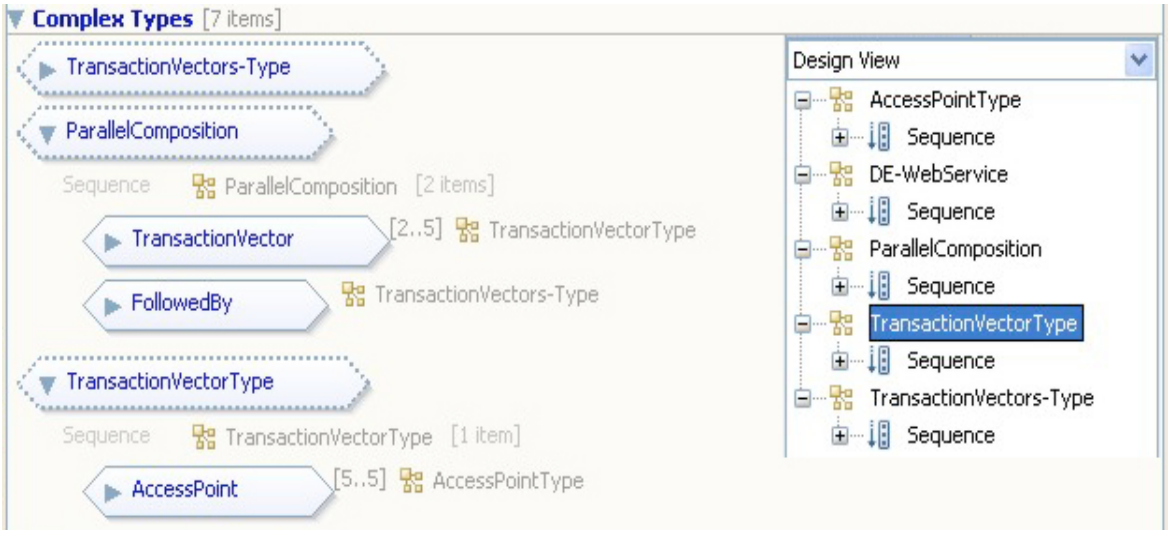


Fig. 3-6 Parallel service composition (xml schema presentation)

3.2.4. Order structure and service dependencies

Based on the prefix ordering between transaction vectors in the set V we may also model a choice between actions. That is, actions which are mutually exclusive in that occurrence of one excludes occurrence of the other.

In discussing concurrent actions in a long-running transaction, we saw that the two incomparable transaction vectors represent concurrent behaviour. The vector they both cover is in fact their greatest lower bound and is obtained by applying the operation *glb()* given earlier. The fact the two incomparable vectors represent concurrent actions is only because they are bounded above in the set (by the transaction vector which is their *lub()* and is sitting on top of the lozenge). Whenever this latter requirement does not hold we may talk about events in conflict.

It might be instructive to make the distinction in terms of pictures and associated Hasse diagrams. In the diagram of Fig. 3-7, *s1* and *s2* are sequential (*s2* can only be invoked after *s1*) in Fig. 3-7(i) while there is a choice between them (alternative) in Fig. 3-7(ii), and they are concurrent in Fig. 3-7(iii).

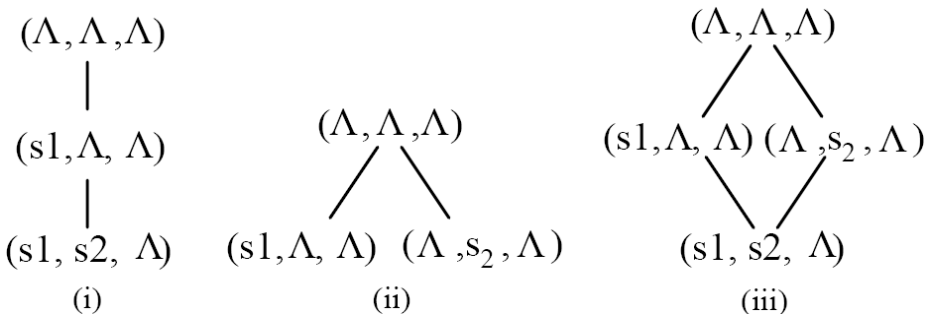


Fig. 3-7 Order structure of transaction vectors

Notice that the set of vectors in (i) does not include (Λ, *s2*, Λ), which means that *s2* never occurs before *s1*; in (ii) it does not include (*s1*, *s2*, Λ) which means there is no valid behaviour of the transaction processing system in which both *s1* and *s2* have taken place; in (iii) it includes all four vectors, which means that *s1*, *s2* and both *s1* and *s2* (simultaneously or at the same time) are all valid observations of the behaviour of the transaction system in which *s1* and *s2* happened concurrently. This is indicated by the familiar lozenge shape that exhibits the characteristic structure of a finite lattice.

In further explanation, the vector sitting at the top of the lozenge is the *glb()* of the two incomparable vectors (*s1*, Λ , Λ) and (Λ, *s2*, Λ) sitting at the middle of the lozenge while the vector at the bottom is their *lub()*. The lozenge as a whole describes that part of behaviour of the transaction in which *s1* and *s2* happened concurrently, as indicated by the vector at its bottom. Further details on how the ordering relations between actions are manifested in the resulting order structure of the resulting set of vectors can be found in [13].

Fig. 3-8 presents the alternative service composition schema. An 'AlternativeComposition' consists of two or more (in fact, up to the number of its access points; here 5) alternative transaction vectors. Notice that in this case there is no 'FollowedBy' element as vectors describing alternative service deployment do not lead to a common vector, i.e. do not have a *lub()* in the formal language.

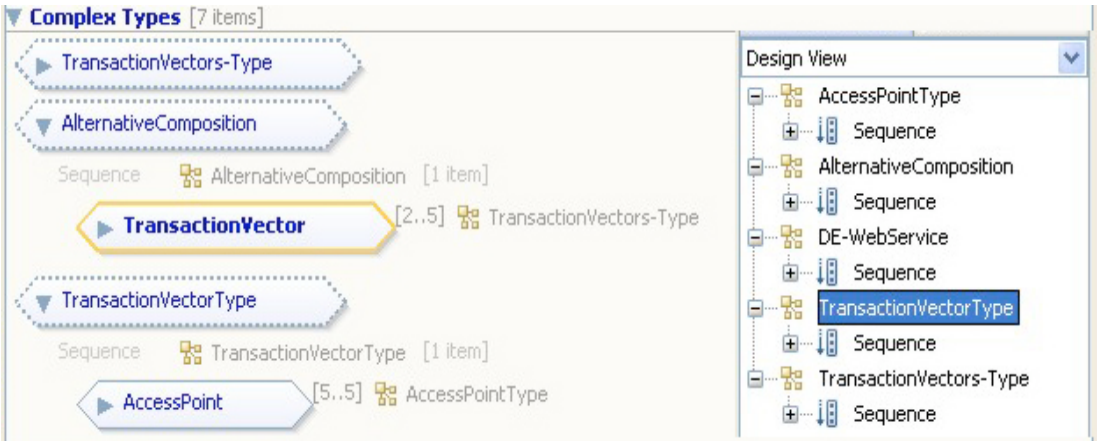


Fig. 3-8. Alternative service composition (xml schema presentation)

Therefore, in our approach given the tree structure of a transaction we may derive a formal description of its intended behaviour, in terms of activations of its sub-transactions and the coordination between them. The resulting behavioural patterns (recall Fig. 3-7) can be analysed before run-time as a means of preventing certain anomalies (such as race conditions) which could result in unexpected behaviour when the transaction actually takes place [13].

3.3- A Scenario with Sequential Service Dependency

We have used JAX-WS 2.1, Java 5, and Netbeans IDE 5.5.1 in our implementation framework. The JAX-WS is a new standard for message passing, it supports both synchronous and asynchronous message passing by the polling model and the call-back model, respectively. In the polling model, the client continuously polls the service response. In the call-back model, the client creates a call-back handler. JAX-WS is shown to perform better than JAX-RPC in certain aspects [14]. Furthermore, JAX-WS supports static and dynamic generation of web service client stubs.

In our implementation, the JAX-WS call-back message passing and dynamic client stub generation are used. The call-back message passing is suitable for asynchronous message passing, which is important for long-running transactions. The dynamic WS client creation enables web service invocation chains. Three participants are required in order to make a service X work: 1) X web service; 2) X TransactionAgent web service; 3) the client stubs for the X web service. The dependencies between web services are captured in an XML file, as discussed before.

From the transaction vector schema, we can create different transaction scenarios (as an XML file). The generated xml file shows the various states of a transaction for a specific scenario. We have designed a software agent coordinator which can perform the coordination of the service invocations as prescribed by the transaction vector language in a fully distributed manner. As a case study we analyse a simple transaction, with a sequential service composition, which involves the interactions shown in the sequence diagram of Fig. 3-9.

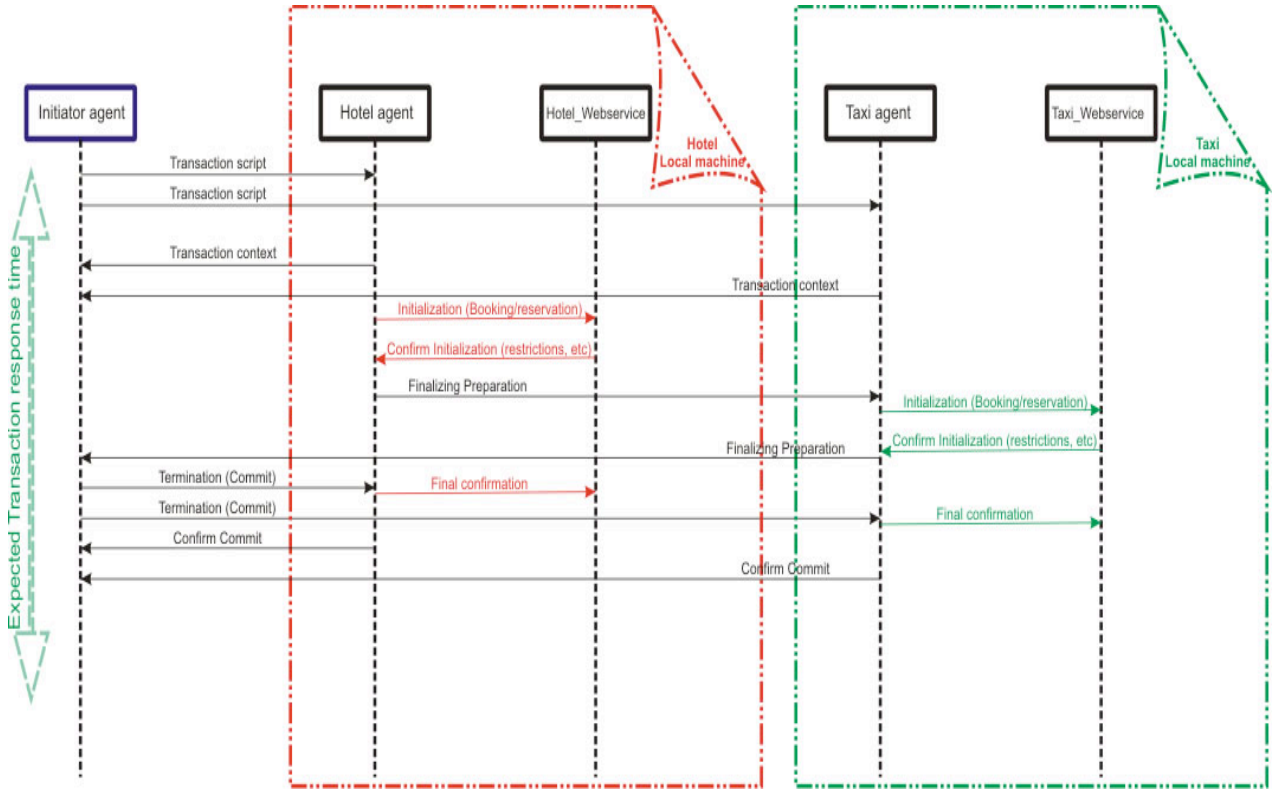


Fig. 3-9 A simple transaction with a sequential service composition

Our simple transaction scenario involves two participants, located at two different SMEs (Hotel and Taxi services). The initiator agent has an AgentHelper that communicates with the TransactionAgents. The transaction involves four players: HotelTransactionAgent (HTA; local coordinator of the Hotel), HotelService (webservice of the hotel), TaxiTransactionAgent (TTA; local coordinator of taxi), and TaxiService (webservice of the taxi). The AgentHelper plays the Initiator role and it starrts the transaction by sending messages (setting the transaction context) to both HTA and TTA. This specifies the first web service should be deployed on the first access point (HTA) and the second will be deployed only after receiving the successful confirmation of the first (sequential service composition).

The following information about the required distributed coordination can be derived: 1) the AgentHelper will know that after it initiates the HTA and TTA, it needs to wait for a response from the TTA for the final preparation status; 2) the HTA will know that it depends on TTA so it needs to send a message to TTA asking for final preparation status; 3) the TTA will know that it gets a request message from the HTA and then sends its response message back to the AgentHelper for the final preparation status.

Figure 3-10 shows the performance analysis of the case study in terms of CPU performance, memory usage, and thread usage when the client performs a commit transaction. The Netbeans 5.5.1 source is available following [9].

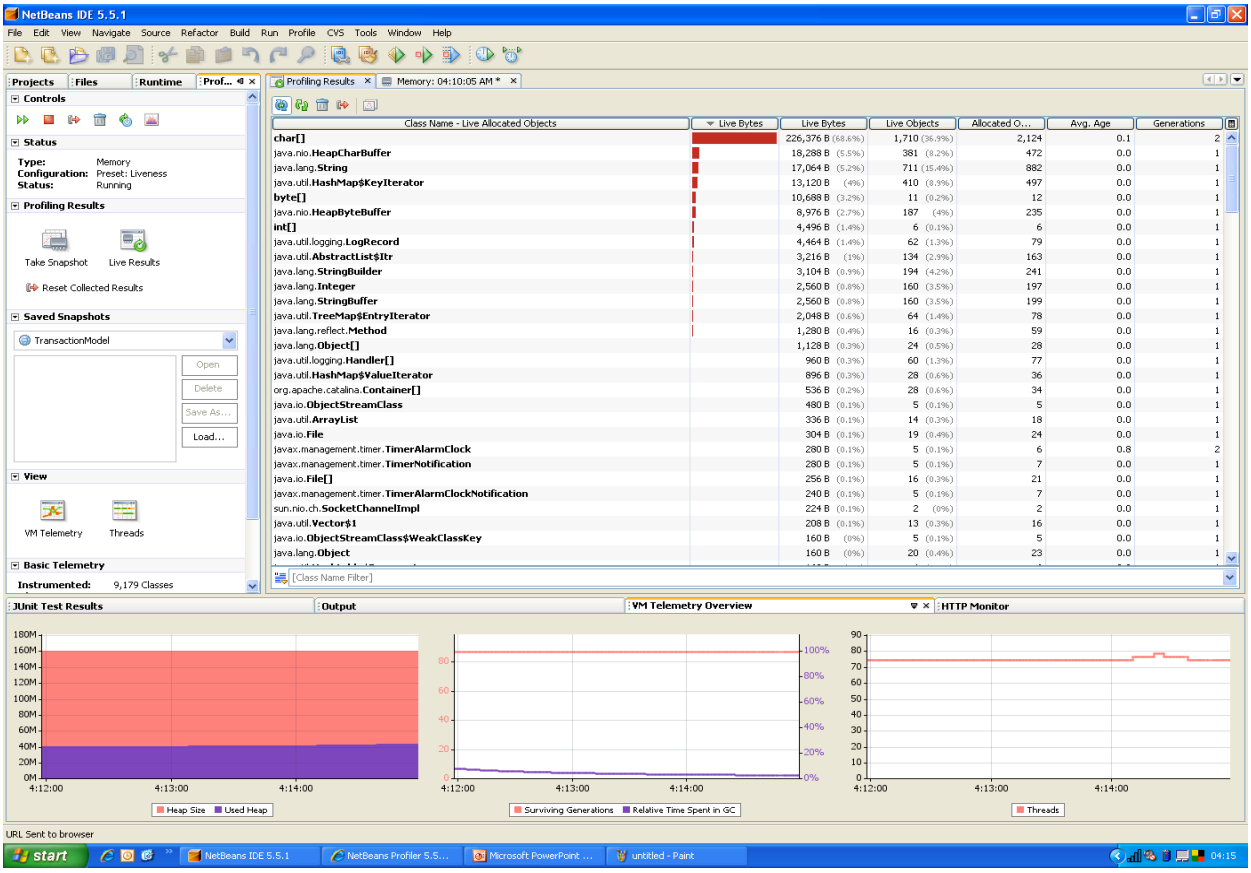


Fig. 3-10 Run-time behaviour of the scenario

3.4- Concluding Note

In this section we have been concerned with the distributed coordination of the service compositions involved in long-running transactions in DEs, and have described schemas which can be used to derive XML representations of the underlying formal model (transaction vectors) and guide the implementation of the proposed transaction model. This was illustrated with a simple example involving sequential service dependencies in a transaction between three participants.

When talking about service composition, in SOA terms but also more generally, we need to take into account dependencies that may arise due to: (i) ordering (of service invocations) and, (ii) data dependencies due to exchange or sharing of results. In this section we have dealt with dependencies that arise due to ordering. In previous work, reported in detail in Deliverable D3.2 of OPAALS [23], and subsequently published in [22], we have described an extended lock scheme that is used to handle data dependencies in a transactional setting with the desired capabilities.

Similarly, dealing with the order of service compositions in long-running transactions comes with a need for coordinating the required service invocations in a principled manner. This is not only relevant for ensuring that the set-up of the transaction includes no more than the desired behavioural scenarios (we have more to say on this in the closing paragraph of this section) but is necessary for the complete specification of a long-running transaction, in terms of expressing both *forward* and *compensating* behaviour. So in addition to modelling the sequences of service invocations required for a successful outcome, it is also necessary to be able to model the sequences of compensating actions required when some forward action of the transaction fails (compensating behaviour). We have not dealt with compensating behaviour in this section, and therefore refer the reader to Deliverable D3.2 of OPAALS [23] which deals with this additional dimension of a multi-service transaction setting and gives a detailed account of how we go about handling failures in our model.

Our model for long-running transactions, to the extend it has been described in this section, provides a way of expressing the service coordination implied in a given transaction along with a schema representation for the formal modelling of such coordination, which can be used to derive XML descriptions of the required orderings on service invocations, the so-called *transaction scripts*. This aspect of the OPAALS distributed transaction model can be extended in interesting ways. In particular, we have been concerned with the refining the initial specification of a transaction, which may be given as a UML model of service interaction scenarios, to ensure that the actual order in which services are invoked respects the required orderings of service invocations in the specification. Our ideas on the gradual elaboration of behavioural scenarios in long-running transactions can be found in [21].

3.5- References (for chapter 3)

[1] M. P. Papazoglou. Service-Oriented Computing: Concepts, Characteristics and Directions. In *Proc. WISE'03*, IEEE, pp. 3-12, 2003.
[2] M. P. Papazoglou, P. Traverso, S. Dustdar et al. Service-Oriented Computing Roadmap. In *Dagstuhl Seminar Proc. 05462, Service-Orienrted Computing (SOC)*, pp. 1-29, 2006.
[3] H. Garcia-Molina and K. Salem. Sagas. In *ACM SIGMOD*, pp. 249-259, 1987.
[4] L.F. Cabrera, G. Copeland, J. Johnson and D. Langworthy. Coordinating Web Services Activities with WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity. January 2004. Available: http://msdn.microsoft.com/webservices/default.aspx
[5] L.F. Cabrera, G. Copeland, W. Cox et al. Web Services Business Activity Framework (WS-BusinessActivity). August 2005. Available http://www128.ibm.com/developerworks/webservices
[6] L.F. Cabrera, G. Copeland, M. Feingold et al. Web Services Coordination (WS-Coordination). August 2005. Available http://www-128 .ibm.com/developerworks/webservices/library/specification/ws-tx

[7] A. Razavi, S. Moschoyiannis, P. Krause. A Coordination Model for Distributed Transactions in Digital Ecosystems. In *IEEE Digital Ecosystems and Technologies* (*IEEE-DEST'07*), 2007.
[8] A. Razavi, P. J .Krause and S. K. Moschoyiannis. Digital Business Ecosystem (DBE) Project Deliverable D24.5: DBE Distributed Transaction Model, 2006.
[9] http://www.computing.surrey.ac.uk/personal/st/S.Moschoyiannis/trnscripts,html
[10] M. W. Shields. *Semantics of Parallelism*. Springer-Verlag, 1997.
[11] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*, Cambridge University Press, 1990.
[12] A. Mazurkiewicz. Basic Notions of Trace Theory. In *Proc. Linear time, Branching Time and Partial Orders in Logics and Models of Concurrency*, LNCS, 354, pp. 285-363, Springer, 1988.
[13] S. Moschoyiannis. Specification and Analysis of Component-Based Software in a True-Concurrent Setting. PhD Thesis, University of Surrey, 2005.
[14] http://java.sun.com/developer/technicalArticles/WebServices/high_performance/
[15] M. Butler, A.C.R. Hoare, C. Ferreira. Trace Semantics for Long-Running Transactions. In *25 Years of CSP*, LNCS 3525, pp. 133-150, 2005.
[16] Bruni, R.; Melgratti, H.; Montanari, U.; Theoretical Foundations for Compensations in Flow Composition Languages. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (*POPL 2005*), pp. 209-220, ACM Press, 2005.
[17] S.Moschoyianis, M.W. Shields. A Set-Theoretic Framework for Composition. *Fundamenta Informaticae* 59(4):373-396, 2004.
[18] A. Razavi, S. Moschoyiannis and P. J. Krause "Preliminary Architecture for Autopoietic P2P Network focusing on Hierarchical Super-Peers, Birth and Growth Models." OPAALS project Deliverable D3.1, 2007 - available at: http://files.opaals.org/OPAALS
[19] Digital Business Ecosystem (DBE) Project, EU-FP6 Programme, available at: http://www.digital-ecosystem.org
[20] S. Moschoyiannis, A. Razavi, Y. Zheng and P. Krause. On Long-running Transactions: semantics, schemas, implementation. In 2nd *IEEE Int'l Conference on Digital Ecosystems and Technologies* (*IEEE-DEST'08*), IEEE Computer Society, 2008.
[21] S. Moschoyiannis, A. Razavi and P. Krause. Transaction Scripts: Making Implicit Scenarios Explicit. In *ETAPS 2008 – Formal Foundations of Embedded Software and Component-Based Architectures* (*FESCA'08*), Electronic Notes in Theoretical Computer Science, Elsevier, 2008. *To appear*
[22] A. Razavi, S. Moschoyiannis and P. J. Krause. Concurrency Control and Recovery Management for Open e-Business Transactions. In *Proc. of Communicating Process Architectures* (*CPA'07*), pp. 267-285, IOS Press, 2007.
[23] A. Razavi, S. Moschoyiannis and P. J. Krause. Report on formal analysis of autopoietic P2P network together with predictions of performance. OPAALS Project Deliverable D3.2 from Phase 1, 2007 - available at: http://files.opaals.org/OPAALS

4. What, not how: a generative approach to service composition (UniS: A. Marinos, P. Krause)

4.1- Introduction

In many ways, service composition has been the ultimate promise of service oriented computing. The ability to pick out services from the open web and connect them arbitrarily could revolutionize the way business is conducted. However, the hidden complexities and barriers to entry for both consumers and producers have so far prevented this promise from materializing. The current approaches to service composition, bound by the complexity of the WS-* stack, have prevented business people from utilizing it for exposing services but also for composing services exposed by others. However, ideas not previously examined are coming to the forefront, either new or not previously considered. This chapter attempts to explore how these new approaches that touch on different aspects of service composition and can help make it available to a much broader audience. This chapter is structured as follows: Section 4.2 examines the various aspects of service composition and the alternative approaches that can be leveraged. Section 4.3 sets the foundations for an architecture that utilizes these alternatives to achieve the pre-existing goals. Section 4.4 goes into details of how such a system would be implemented while section 4.5 offers a case study that illustrates the concepts that were presented in sections 4.3 and 4.4. Section 4.6 gathers the future work that still remains and offers some concluding remarks.

4.2- The Building Blocks

4.2.1. Service Composition

Service composition is the process of combining atomic services in order to achieve a composite goal. Service composition is separated into manual composition and automated composition. In manual composition, the services are positioned in a workflow by the user whereas in automated composition the user defines the goal of the service composition and depends on a software tool to define the most suitable way to achieve this goal. According to [2], automated composition can also be subdivided into three categories.

The first is ‘Fulfilling Preconditions’ in which pre-existing services are combined in UNIX pipeline fashion to fulfill the requirements of a service that does not exist in atomic form. For example a .doc to .pdf converter and a .pdf printer can be combined to emulate a .doc printer.

The second is ‘Generating Multiple Effects’ in which a number of services should be executed to produce separate but interrelated outcomes. A typical example of this type of composition is the travel scenario where flight and hotel can be booked independently but must be coordinated for their result to be useful.

Finally, a type of composition called ‘Dealing with Missing Knowledge’ is defined, where for instance a list of metro stations may be combined with a list of hotel addresses to identify hotels near metro stations. In this type of composition additional information sources are queried and the results are combined with the results from a service provider to satisfy more complicated queries.

While these types of service composition can be independent, there are problem domains which require a combination of approach. In this regard, the three types can be considered building blocks for a complete service composition system.

4.2.2. Declarative versus Imperative Programming

Two major approaches to computer programming are the declarative and the imperative approaches. In order for their differences to be understood, it is important to separate the 'what' from the 'how' of a solution to a computing problem. The 'what' refers to the properties that a solution must possess whereas the 'how' refers to the steps followed to achieve the required solution. Declarative programming focuses on specifying the 'what' and depending on a software tool to decide which is the most appropriate way of reaching the goal. An example of a declarative language is the well-known SQL which specifies properties of data but not the way to retrieve it, which is left to the database implementation.

Imperative programming focuses on the 'how'; bypassing the need to define the properties of the required solution since the programmer can guarantee the desired properties by directly controlling the algorithm. Java is considered an imperative language, although in later versions, declarative elements have appeared either in the language itself or in libraries designed for it.

In the domain of service composition it is clear that there is a strong correlation between automated composition and declarative programming. Similarly, manual composition can be thought of as an application of imperative programming. It is important to be aware of the connection between programming paradigm and service composition strategy when selecting the appropriate method of interfacing with the user.

4.2.3. Expressing Requirements

In efforts related to service composition such as [6], various languages such as OWL-S have been used to describe the user's desired goal, usually serialized in XML. These languages require training to read and write but provide exact semantics to machines so that they can be directly executed.

A new and promising approach to human-computer interfacing is OMG standard Semantics of Business Vocabulary and Business Rules (SBVR [3]). As defined by [2], "SBVR provides a way to capture specifications in natural language and represent them in formal logic so they can be machine-processed". This allows users to be able to verify the requested service composition by directly reading the structured natural language used by SBVR which can then be parsed and executed by a machine.

Since SBVR is a way to capture specifications, there is no technical barrier to following any programming style. Specifically, one could conceivably use it to specify the structure of a process therefore conforming to an imperative programming style. However, the SBVR specification [3], quoting the Business rules manifesto [4] states:

*"**Separate From Processes, Not Contained In Them**. Rules apply across processes and procedures. There should be one cohesive body of rules, enforced consistently across all relevant areas of business activity.*

***Declarative, Not Procedural**. Rules should be expressed declaratively in natural-language sentences for the business audience. A rule is distinct from any enforcement defined for it. A rule and its enforcement are separate concerns."*

So, while SBVR can be used in an imperative style, the full benefits of the Business Rules Approach that spawned it can only be attained when it is used declaratively. Work on expressing information systems declaratively with the use of business rules has been presented in [10]. This work predates SBVR but provides essential guidelines to its utilization in an information system.

4.2.4. Distributed Computing Architectural Style

Service composition approaches in literature assume a foundation of Web Services built on the WS-* stack including specifications such as SOAP, WSDL, XML Schema and others. It is to be expected that this paradigm is used in literature given the activity around it in industry and standards organizations in the

past few years. Essentially the WS-* stack represents RPC-style interaction tunneled through the HTTP protocol as a transport.

However observation of uptake in the wild suggests that a different paradigm is gaining ground and recently industry seems to be noting this. The APIs of major web applications from Google[13], Amazon[16] and many others are built using a different architectural style termed REpresentational State Transfer (REST). Even WS-* heavyweights such as Microsoft are working on projects that utilize REST [12], [14]. Also recently a number of standards have been approved that are compliant with and based on the REST style. [15], [17].

Contrary to what may be assumed, REST is not a new development. It was first identified in 1999 by Roy Fielding in his PhD dissertation [5] as a term to describe the architectural style used by the web. Consequently, HTTP 1.1 was released to better align the web with the principles of REST. While many have since championed REST as a competing web service paradigm to the WS-* stack, it has only recently begun to be more seriously considered with the publication of works such as [7] and the apparent lack of expected adoption for WS-* technologies outside the corporate firewall.

The main concept of REST is the resource as a document that is identified by a URI, a uniform resource identifier. Resources are to be accessed by a universal interface of well defined methods that should be resource-agnostic and therefore have the same, standard effect on all resources. In the case of HTTP 1.1, the methods include GET, PUT, POST and DELETE [18]. Other protocols such as WebDAV define methods such as LOCK and UNLOCK, suitable for transactions on resources. This spartan interface is in contrast with the WS-* approach of defining a multitude of unique methods, one for each use case to be executed.

In the context of service composition, the separation between verbs (methods) and nouns (resources) encourages a more declarative style of expression where the user is concerned with constraining the outcomes of the service composition rather than the providers of the services or orchestrating procedure calls. REST also yields a more loose-coupled service composition since service providers can be alternated based on resource provided rather than interface compatibility.

4.3- Design and Architecture

4.3.1. Requirements

Service composition within a Digital Ecosystem requires focus on the items to be composed. Additionally, it is necessary to approach the business users with a perspective familiar to their pre-existing mindset. This requires minimizing the learning curve or if possible removing it altogether. As the main focus is on Small and Medium Enterprises, the barrier to implementation of a service on the ecosystem should also be considered.

4.3.2. Design Approach

Since we are operating within the context of a digital ecosystem, our initial focus is on ‘Generating Multiple Effects’ type of service composition that can handle use cases similar to the travel scenario. We believe that ‘Fulfilling Preconditions’ and ‘Dealing with Missing Knowledge’ types of composition can be incrementally added and are currently considered parts of future work.

A declarative programming paradigm is selected in order to move the burden of implementation to the providers of the platform rather than the business users. In this way the business user can express requirements by stating intent rather than workflow. SBVR complements the declarative approach and lowers the barrier to entry even further compared to XML-based languages.

A RESTful architecture brings improved loose-coupling, simpler implementation and leverages the pre-existing infrastructure of the web. A REST web service can also be naturally utilized as part of a businesses web presence by acting as the server to an Ajax in-browser client therefore encouraging reuse and reducing implementation overhead. Additionally, resource-orientation is natural for service composition within a Digital Business Ecosystem since the consumer is not interested on the provider but in the resource (product) provided and it would be advantageous for the underlying architecture to also reflect this.

At this point, it is important to make a technical distinction in the types of resources that can participate in a RESTful service composition. On the one hand there are bounded resources. These resources exist before being requested for by the user and therefore lists of available resources can be made to any query. For example a seat on a specific flight exists before being requested by the user. On the other hand, there are resources that are created specifically based on a user's request. These resources are called unbounded resources. It is important to note that these resources cannot be queried through a generic query but have to be requested on a one by one basis. An example of such a resource is a taxi reservation. If one were to ask what taxi reservations could be made during a specific day, the list would be infinitely long due to the granularity of such a service, and therefore unusable.

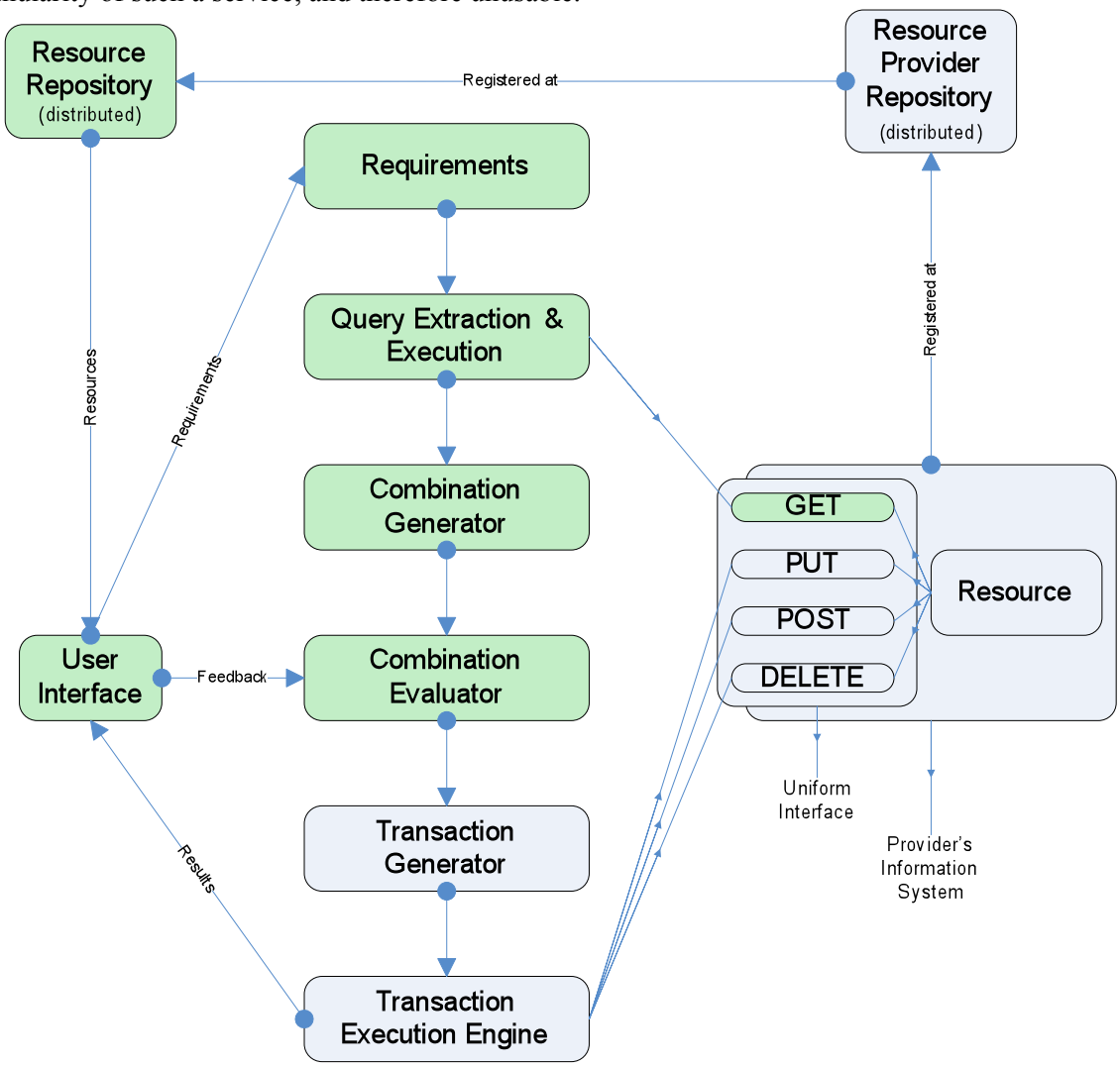


Fig. 4-1: Overview of the Service Generation Process

4.4- Implementation Principles

Figure 4-1 above provides a schematic overview of the proposed model for generating a transaction that responds to a consumer request. This section provides more detail of the individual components in this model.

4.4.1. Service Description

While RESTful web services have had uptake in practice, a suitable description language has not yet been standardized. A notable effort is Web Application Description Language [9] developed by SUN which is purpose built for describing RESTful web services but not yet mature. An alternative is WSDL 2.0 [19] which has been extended to express RESTful services but it's WS-* rooted complexity is a significant drawback. Additionally, expressivity provided by these languages is limited to variable types. Additional implementation information is expressed as text to be read by a developer. Leveraging SBVR as the basis of a service description language should provide sufficient capabilities for integration with minimal human intervention by reducing the complexity level and learning curve presented by these languages. Additionally SBVR can provide integration advantages when used as a language to express both services and requests on these services. Service description in a RESTful SOA is different to a service description in the WS-* world due to the fact that REST is constrained by the uniform interface. In this regard, the typical function of the WSDL file to describe the methods and arguments relevant to a service becomes redundant. What is required of a service description in REST is to identify the resources available within a service, the media types or schemas that can be used by an automated agent to process the information returned, and also the subset of the standard verbs that the resource will respond to.

4.4.2. Repository

A service repository in a RESTful architecture has not yet been considered in literature. This seems reasonable considering the poor uptake of UDDI and the usage patterns associated with web services on the public internet. It is however useful to consider it within the context of a Digital Business Ecosystem. What is needed from the repository is to have semantics for all the resource types used within the ecosystem and the ability for service providers to register as providers of a specific resource. Any provider should be able to create a new resource type however it is expected that market forces will lead to demand-driven standardization of resource types. In fact it should be more feasible for a business owner to register as a provider of the competition's resource than it is to implement the same service interface as would be required in the WS-* approach. Additionally, the repository itself should be built with distribution in mind so as to avoid single points of failure. However, the distribution architecture of the repository is beyond the scope of this chapter.

4.4.3. Requirements Expression

Since our initial focus is on the 'Generating Multiple Effects' type of service composition, we should use SBVR as a means of expressing the desired effects and correlations between them. Within a RESTful architecture this translates to resources and constraints on their attributes. Therefore a two step process is needed. First the user selects the desired resources from the repository. The interfaces of the services are expressed in SBVR and from them an aggregated SBVR vocabulary for the service composition is generated. Then, based on this vocabulary, rules are written to express what combinations of these resources are acceptable with both absolute and relative constraints. Absolute constraints refer to properties of the combination itself whereas relative constraints refer to properties of a given combination of resources in comparison to all other available combinations.

4.4.4. Combination Generation

According to the vocabulary and rules that express the request, the combination generation algorithm should query the relevant providers and determine the combinations that satisfy the absolute constraints.

Methods of RESTful querying are already implemented in Microsoft Astoria and Google's GData [11] which can serve as a reference.

Determining the appropriate combinations requires identifying a suitable algorithm. Backtracking may fit the requirements; however there should probably be a more efficient way to search the solution space for matches. It is important to note here that bounded and unbounded resources should be treated at different phases due to their different nature. At first, range queries are made to all bounded resources once and viable combinations of the results generated. Afterwards, the unbounded resources are queried repeatedly, once for each generated combination of bounded resources. At the end of this process, a list of feasible combinations should be produced.

4.4.5. Combination Evaluation

After the suitable combinations are determined, they can be evaluated either manually or automatically. Automated evaluation depends on relative constraints entered during requirements expression whereas manual evaluation returns the combinations to the user for arbitrary ordering. This is of course preferred because in many cases users will have additional implicit requirements not stated which will cause them to alter the results of the automated evaluation. The two approaches can be combined by using automated evaluation as a step before manual evaluation therefore easing the user's work. However, if a composition is to be exposed as a service itself, manual evaluation is not feasible, since the user will be represented by a machine which will only be aware of the initial requirements and will not be able to provide more accurate insights relevant to processing the returned results. Manual evaluation remains a valuable option where the ability to apply it exists.

4.4.6. Strategies for generating the transaction tree

In order to generate a transaction tree from a set of combinations there are a number of strategies that must be followed. The strategies differ in the complexity of the transaction tree produced and therefore the amount of intelligence of the transaction model that they utilize. As more complex strategies are explored, it is made clear that a balance must be struck between the decisions that the transaction can take at runtime and the decisions that the user can take at design-time. Deciding this will determine where the barrier between the service composition and the transaction execution phases will be set.

Combination-centric transaction

The easiest way to create a transaction from a set of combinations of resources is simply to consider each combination a sub-transaction, coordinated in parallel and then use a serial alternative coordinator for the entire transaction, ordering the sub-transactions in order of preference of the combinations which they represent. Following this strategy gives us an elegant way to derive a transaction from a set of combinations.

However this simplicity may prove problematic as resources that may appear in multiple combinations may have to be locked and unlocked more than once as the transaction proceeds therefore creating inefficiencies for the provider and risk having a resource made unavailable in the time between unlocking and locking again. Additionally these repeated calls to resources may have performance implications that again may result in unavailability of other resources.

Factoring out overlapping services

In order to deal with this problem, we have to utilize a different strategy that allows us to factor out overlapping resources into separate sub-transactions, therefore avoiding the possibility of multiple

interactions with the same resource. The high level coordinator for such a transaction will be a serial coordinator. Sub-transactions will have similar structure to a combination-centric transaction without overlap. In order to achieve such a structure, we will make use of data driven coordinators to allow sub-transactions to communicate their results to one another. This will allow us to ensure that in any case, the result of a successful transaction is one of the initial combinations, which is the main goal of any transaction tree that is generated.

Deferring decisions

Having established the possibility of pursuing each type of resource as a separate sub-transaction, a new possibility is made available. The decision for the exact resource that will satisfy the requirement for an unbounded resource can be rolled over to the transaction execution phase. It is therefore possible to avoid rejection of a combination simply because an unbounded resource that was available at the time of querying is no longer available at the time of execution of the transaction. This would increase the probability that a transaction will commit successfully. In order to achieve a transaction tree that facilitates this functionality, we would have to extend the structure presented in the "Factoring out overlapping services" strategy by adding the ability for abstract resources to be added as separate sub-transactions. These sub-transactions would be connected to others with the use of data driven coordinators. Once the data has been delivered to the sub-transaction, the provider is queried and a suitable resource is selected.

Implementing this strategy requires modifying the process of service composition to allow creation of less specified results. In this regard it is not directly compatible with the service composition workflow as it stands but is presented as a future direction where the transaction model is empowered with greater decision power.

4.4.7. Adjusting the transaction model for a REST environment

For the transaction model to be used as part of the work on service composition, it is required that it is adapted to work in a RESTful environment. The main area that is affected is the interactions with the services and more specifically the locking system. These interactions must be made to work with a more narrow definition of service that consists of accessing a resource through the uniform interface. The following paragraphs describe two alternative approaches to creating a protocol that allows interactions with resources while being tailored for a specific application domain such as transaction execution.

Replacing or extending the HTTP protocol

The REST architectural style has been connected with the World Wide Web and the HTTP protocol. However the HTTP protocol is only one implementation of the REST style. It is therefore possible to imagine other protocols completely separate from HTTP that implement the REST style. The most significant difference would be in the methods that make up the uniform interface. A variant of this approach would be to extend the HTTP protocol with additional methods that help achieve the goals of the new protocol. A good implementation of this approach is the WebDAV protocol that adds methods such as PROPFIND, PROPGET, LOCK, and UNLOCK. These are used to perform transactional operations on resources. In fact, the WebDAV protocol [17] has been created to emulate a file system and therefore gives valuable insights on how to implement transactions over REST.

The weakness of this approach is that while it maintains the theoretical advantages of the REST style, it also foregoes the practical advantages that arise from the widespread adoption of the HTTP protocol as a common language. For this reason it is useful to consider other alternatives that while not as straightforward, retain both the theoretical advantages of REST and the practical advantages of HTTP adoption.

Using patterns

To avoid foregoing the HTTP protocol while still achieving the desired behaviors on both the client and server side it is necessary to adopt a number of conventions. These conventions will provide guidance to the interacting parties on how to guide their interaction within the HTTP protocol in order to achieve the desired results. Regarding the locking of resources, [17] mentions the possibility of representing locks as separate resources and using the uniform interface to manipulate them. For example if a resource can be found at www.example.org/clients/42 then a pattern could specify that the lock for the specified resource would be found at www.example.org/clients/42/lock. Then the client could use the PUT, GET and DELETE methods to set, read and remove a lock respectively. This functionality would also require an XML schema for a lock that is created in accordance with the needs of the transaction model and is understood by all participants in the transaction. Another more advanced possibility for locking would be to replace a single lock with a collection of locks for resources that can be locked in parts and therefore have multiple locks. For example a document could have multiple locks assigned to several of its paragraphs when multiple editors are editing different parts. The client could then use GET to read the collection (e.g. www.example.org/documents/42/locks), retrieve each lock if it exists (e.g. www.example.org/documents/42/locks/1), use POST to place a new lock, and DELETE to remove the lock once it is no longer required. The benefit of using HTTP as the basis for the implementation is that all the infrastructure in place, including hardware such as firewalls and routers but also software such as programming libraries, web servers and even browsers that understand the semantics of HTTP are already in place and can be used with the therefore greatly reducing the amount of effort required to implement such a protocol.

The elements presented here are sufficient to implement a simple locking system on top of HTTP but a complete transaction model with all the features required for a viable digital ecosystem a more effort is required of which the preceding is only a part. Work on this is currently in progress.

4.5- Case Study

In order to illustrate the concepts presented above, this section includes a case study. The case study is based on the well-known travel scenario where a user is trying to fulfil a request for travel arrangements.

4.5.1. Service Description

Each resource in the ecosystem is described by an associated set of vocabulary and rules that serves as a service description for each of the providers of the resource. For example the Flight resource could be described with the statements in Figure 4-2. As seen in the example, a resource could also be constrained by rules.

1. **Flight** *has* **Departure Date/Time**
2. **Flight** *has* **Arrival Date/Time**
3. **Flight** *is from* **Departure Airport**
4. **Flight** *is to* **Arrival Airport**
5. **Airport** *is in* **location**
6. It is obligatory that the **Departure Date/Time** *of* the **Flight** *is before* the **Arrival Date/Time** *of* the **Flight**

Figure 4-2. Example of a resource description in SBVR.

4.5.2. Service Selection

As the user selects the resources that are added to the composition, the associated vocabularies and rules are also added to the service requirements so that the rules about the service requirements themselves can be written using them.

4.5.3. User Requirements

Except for the vocabularies and rules related to the resources to be composed, a user's requirements also contain rules that link the service composition to each resource and also constrain the resources and service composition according to the need of the users. In our case study, the user's requirements are depicted in Figure 4-3. In the figure, statements can be seen connecting the resources such as 'Departure flight', 'Return Flight' and 'Taxi booking' to the service composition and to each other. While statements 1-20 may seem tedious, they do offer a description of the service that can be verified by an untrained user, and can also be reused as a resource themselves. The most interesting statements are the ones from 21 onward that build on the previous ones to express complex constraints on the length and cost of the travel arrangement. It can be thought that a company may have Travel Arrangement service composition templates ready for its employees who then only cave to add the rules relevant to their specific instance of the travel arrangement and execute the service composition as is.

a. Vocabulary

1. Travel Arrangement *is* a Service Composition
2. Travel Arrangement *contains* a Flight *called* Departure Flight
3. Travel Arrangement *contains* a Flight *called* Return Flight
4. Travel Arrangement *contains* a Hotel Booking
5. Travel Arrangement *contains* a Taxi Booking
6. Travel Arrangement *has* a date *called* Departure Date
7. Travel Arrangement *has* a date *called* Return Date
8. Travel Arrangement *has* a time period
9. Time period *extends between* the departure date *and* the return date.
10. Travel Arrangement *has* a location *called* home location
11. Travel Arrangement *has* a location *called* destination location
12. Travel Arrangement *has* an amount *called* total cost

b. Rules

13. It is obligatory that the departure date *of* the departure flight *is same as* the departure date *of* the travel arrangement.
14. It is obligatory that the return date *of* the return flight *is same as* the return date of the travel arrangement.
15. It is obligatory that the departure flight *is from* an airport that *is in* the home location.
16. It is obligatory that the departure flight *is to* an airport that *is in* the destination location.
17. It is obligatory that the return flight *is from* an airport that *is in* the destination location.
18. It is obligatory that the return flight *is to* an airport that *is in* the home location.
19. It is obligatory that the pickup location *of* the taxi booking *is at* the destination airport *of* the departure flight.

20. It is obligatory that the drop off location *of* the taxi booking *is at* the location *of* the hotel booking
21. It is obligatory that the home location *of* the travel arrangement *is* London
22. It is obligatory that the destination location *of* the travel arrangement *is* Athens
23. It is obligatory that the pick up time *of* the taxi booking *is* 30 minutes *after* the landing time *of* the departure flight
24. It is obligatory that the departure date *of* the travel arrangement *is between* December 1, 2007 *and* December 3, 2007.
25. It is obligatory that the time period *of* the travel arrangement *is between* 16 and 18 days.
26. It is obligatory that the total cost *of* the travel arrangement *is minimal*.
27. It is obligatory that the total cost *of* the travel arrangement *is less than* £850

Figure 4-3. Example of a user request in SBVR.

4.5.4. Extracted Queries

Once the requirements of the user have been specified, a software system should take over and attempt to fulfill the requirements of the user. The first step in executing the service composition is to query the bounded resources. This implies that a standard RESTful querying mechanism exists within the ecosystem so the user does not have to specify something in her request. In our case study, the query to one provider for the departure flight would be similar to the following:

GET http://rest.easyflight.com/flights[From:London][To:Athens][date:2007/12/01…2007/12/03]

The GET method is the standard method invoked in HTTP to retrieve information from a service. The query URL is constructed using similar principles as the ones found in Microsoft Project Astoria and Google's GData.

4.5.5. Results

Each query that is executed returns a set of results in plain XML without the use of SOAP envelopes. An example of the content of the results produced by a service query can be found in figure 3.

4.5.6. Combinations

Once all the queries on the bounded resources have been executed, The results are then combined according to the user's requirements to create fitting combinations. Once the combinations are in place, unbounded resources such as the 'Taxi Booking' are queried with the specific data of each combination and the results are inserted to the combinations to complete them.

4.5.7. Evaluation

When the combinations are complete they have to be evaluated in order for the most suitable ones to be selected for execution. Evaluations can have two phases: automated and manual evaluation. Automated evaluation is based on relative and absolute criteria entered along with the requirements. In our case study, rules 25, 26 and 27 are criteria that can help select the most suitable combinations according to the length of the time period and the total cost of each service composition. Application of these criteria falls within the automated evaluation of the generated combinations. However, if the user is interacting with the composition process directly through some user interface, the opportunity for manual evaluation as an additional step exists. The result of combination evaluation is not a single combination as may be expected, because there is no guarantee that the resources that comprise a combination will be available when the

transaction is executed. For this reason we aim for redundancy and keep more combinations, as can be seen in Figure 4.

4.5.8. Generated Transaction Tree

Once the combinations to be executed have been determined, they have to be converted into a form that can be executed by the transaction model. This can be achieved by merging the combinations into a single transaction tree. The transaction tree can be represented with the notation presented in [8]. While the easiest way to do this is to utilize a serial alternative coordinator at the top level and connect to it the different combinations as separate sub-trees, this foregoes a lot of potential for optimization. Ideally, the transaction tree should utilize similarities between the combinations to create more optimal transaction trees that contain fewer leaf nodes and therefore avoid unnecessary service invocations.

EasyFlight.com						
Flight No.	Departure Date/Time (Local)	Departure Airport	Arrival Date/Time (Local)	Arrival Airport	Seat Class	Price
EZF 5243	2007/12/01 06:50	LGW	2007/12/01 08:50	ATH	B	150£
EZF 5248	2007/12/02 12:40	LGW	2007/12/01 14:40	ATH	B	170£
EZF 5242	2007/12/02 17:50	LGW	2007/12/01 19:50	ATH	A	220£
EZF 5243	2007/12/03 06:50	LGW	2007/12/01 08:50	ATH	C	130£

Figure 4-4. Example of results returned by a service.

Position	Departure Flight	Taxi Booking	Hotel Booking	Return Flight	Composition Properties
1	**Departure Date/Time**: 2007/12/03 06:50 **Departure Airport**: LGW **Arrival Date/Time**: 2007/12/03 08:50 **Arrival Airport**: ATH **Seat Class**: C **Price**: £130	**Pickup Location**: Athens Intl. Airport **Pickup Date/Time**: 2007/12/03 09:20 **Destination**: Athens Holiday Inn **Price**: £20	**Location**: Athens **Room Class**: B **Room Capacity**: 1 **Start Date**: 2007/12/03 **End Date**: 2007/12/19 **Price**: £480	**Departure Date/Time**: 2007/12/19 15:50 **Departure Airport**: LGW **Arrival Date/Time**: 2007/12/19 17:50 **Arrival Airport**: ATH **Seat Class**: C **Price**: £170	**Departure Date**: 2007/12/03 **Return Date**: 2007/12/19 **Time Period**: 17 **Home Location**: London **Destination Location**: Athens **Total cost**: £800
2	**Departure Date/Time**: 2007/12/03 06:50 **Departure Airport**: LGW **Arrival Date/Time**: 2007/12/03 08:50 **Arrival Airport**: ATH **Seat Class**: C **Price**: £130	**Pickup Location**: Athens Intl. Airport **Pickup Date/Time**: 2007/12/03 09:20 **Destination**: Athens Hilton **Price**: £20	**Location**: Athens **Room Class**: B **Room Capacity**: 1 **Start Date**: 2007/12/03 **End Date**: 2007/12/18 **Price**: £440	**Departure Date/Time**: 2007/12/18 15:50 **Departure Airport**: LGW **Arrival Date/Time**: 2007/12/18 17:50 **Arrival Airport**: ATH **Seat Class**: C **Price**: £190	**Departure Date**: 2007/12/03 **Return Date**: 2007/12/19 **Time Period**: 16 **Home Location**: London **Destination Location**: Athens **Total cost**: £780
3	**Departure Date/Time**: 2007/12/02 12:40 **Departure Airport**: LGW **Arrival Date/Time**: 2007/12/02 14:40 **Arrival Airport**: ATH **Seat Class**: C **Price**: £170	**Pickup Location**: Athens Intl. Airport **Pickup Date/Time**: 2007/12/03 09:20 **Destination**: Athens Holiday Inn	**Location**: Athens **Room Class**: C **Room Capacity**: 1 **Start Date**: 2007/12/02 **End Date**: 2007/12/19	**Departure Date/Time**: 2007/12/19 15:50 **Departure Airport**: LGW **Arrival Date/Time**: 2007/12/19 17:50 **Arrival Airport**: ATH **Seat Class**: C **Price**: £170	**Departure Date**: 2007/12/03 **Return Date**: 2007/12/19 **Time Period**: 18 **Home Location**: London **Destination Location**: Athens

		Price: £25	**Price**: £475		**Total cost**: £840

Figure 4-5. Combinations generated and ranked.

4.6- Generated Transaction Trees

4.6.1. Mapping the combinations

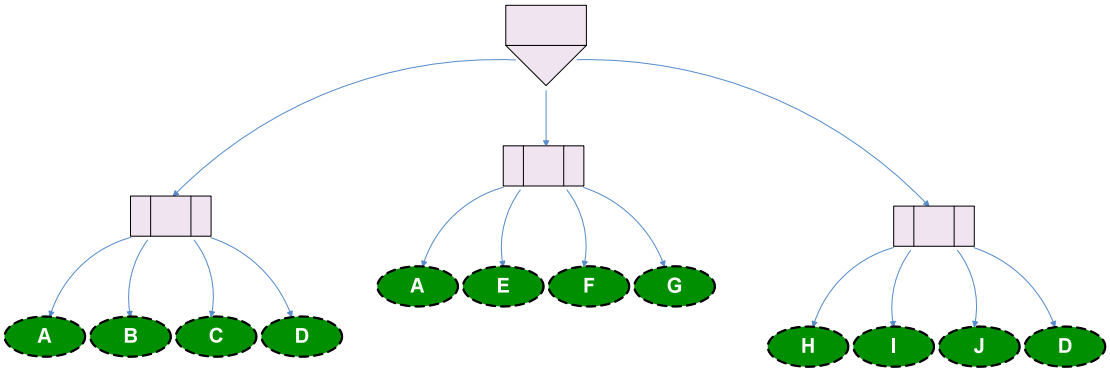
In order to be able to create meaningful transaction trees, we must first map the service instantiations to symbols that can be used in the transaction trees. Replacing symbols for service details in figure 4-5, we arrive at figure 4-6, below. In this figure it is easy to spot the overlapping resources in the various combinations. For example, resource A is used in both combination 1 and 2.

	Flight 1	Taxi	Hotel	Flight 2
Combination 1	A	B	C	D
Combination 2	A	E	F	G
Combination 3	H	I	J	D

Having completed the mapping, we can now progressively apply the transaction tree generation strategies presented.

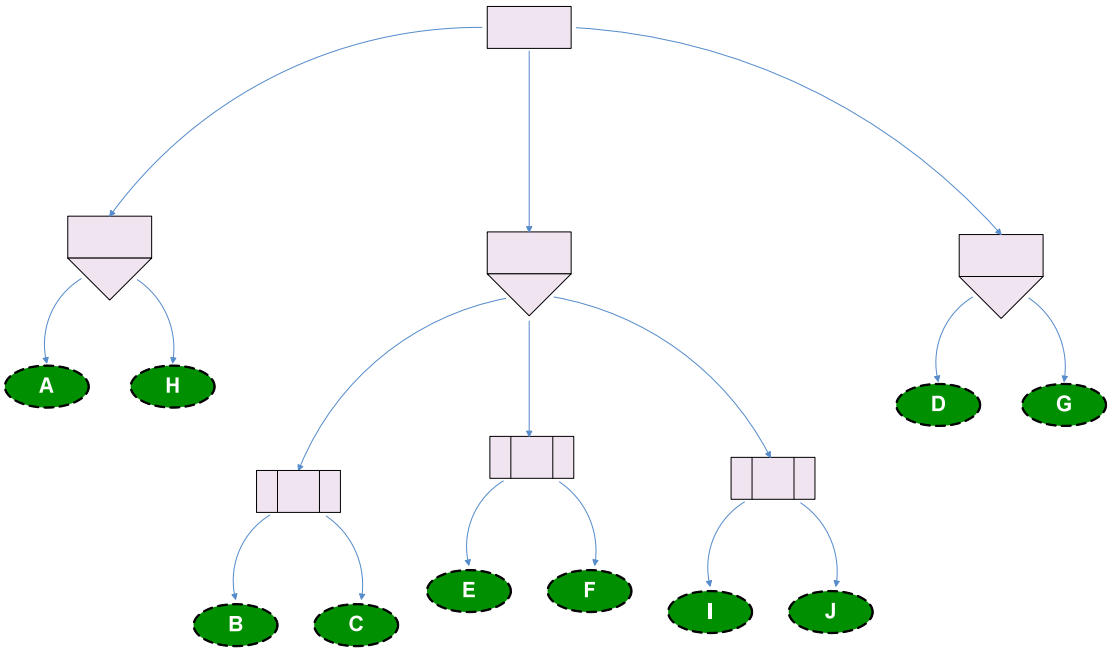
4.6.2. Combination-centric transaction

The first and simplest way to reach an executable transaction tree is to represent each combination as a sub-transaction and join these sub-transactions with a serial alternative coordinator to signify that they are to be attempted in order. However the overlap in resources A and D presents an opportunity for optimization.



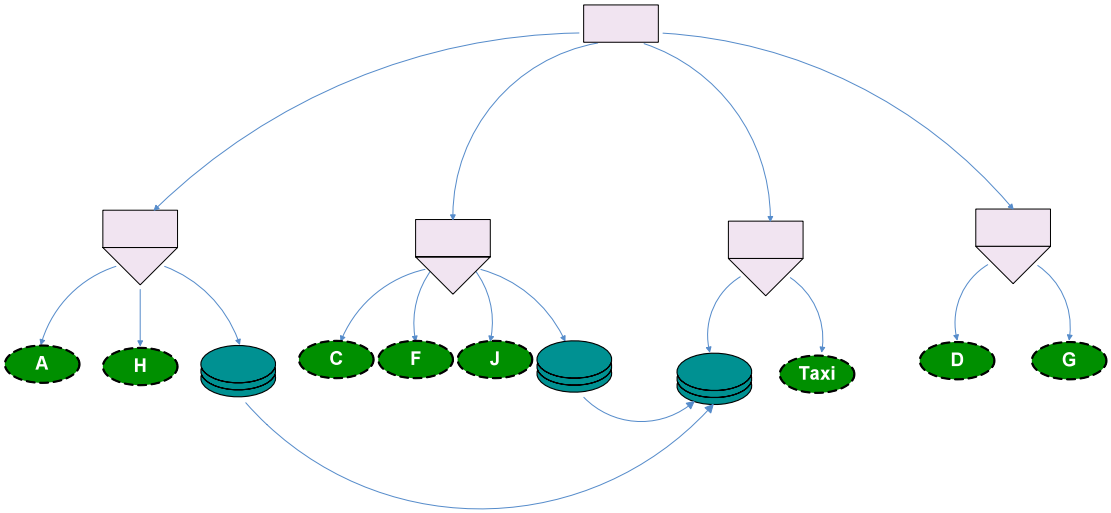
4.6.3. Factoring out overlapping services

By factoring out resources services ‘Flight 1’ and ‘Flight 2’ we are able to avoid the possibility of repeatedly calling the same service therefore improving efficiency and the possibility of success.



4.6.4. Deferring decisions

Since the Taxi service represents an unbounded resource, we are able to defer the decision on the parameters of this resource until the last minute when the flight 1 and hotel sub-transactions are ready to commit. This takes advantage of the expressive power of the transaction model and specifically its data-oriented coordinators to achieve further improvements in efficiency. By this stage in the development of the transaction tree we can see that each service is represented by a sub-transaction and we depend on the expressive capabilities of the transaction model to make sure that a compatible combination of resources is chosen if the transaction successfully commits.



4.7- Conclusion & Future Work

Work still remains to be done, including covering the different types of service composition, other than 'Generating multiple effects'. Additionally, work should be done to examine more "fuzzy" result matching in case the initial queries do not provide results. For instance, a later taxi arrival is preferred to returning no service composition at all. Another large issue is integrating and automating a trust or reputation system so that the providers are selected based not only on criteria relevant to the resource in question, but also their own previous performance. To this end, an appropriate feedback mechanism should be built into the system that allows consumers to rate providers after the resource has been consumed and therefore evaluated. This is essentially a matter of linking our work with the OPAALS work on Accountability, Identity and Trust [20]. Additionally, progress must be made in modifying the transaction model to operate within a RESTful architecture. Again, this is work in hand.

Ideally, users should be aided with a reasoning system being able to manage some common, personalised heuristics. For instance, even human officers in travel agencies sometimes recommend bad travel plans where connecting flights are clearly too tight and require running trough transit halls, the time for taxi trips depend on rush hours, and so forth. This line of thinking could be extended to assert that the task of suggesting transactions (plans) is the job of a general-purpose service composition agent (perhaps as a distributed service of the P2P network), and the task of verifying their sensibility is (quite naturally?) the job of the (trusted) personal helper agent (perhaps as a highly local application). It also illustrates that service composition could benefit from work related to automatic planning and plan verification.

While the work presented in this section is still under development, we feel that we have laid an appropriate foundation for utilizing the emerging technologies such as REST and SBVR within a digital ecosystem for the purpose of declarative service composition. With combining the advantages provided by these technologies, service composition can be made accessible both to service providers and service consumers.

4.8- References (for Chapter 4)

[1] U. Kuster, M. Stern, and B. Konig-Ries, "A Classification of Issues and Approaches in Automatic Service Composition," Intl. Workshop WESC, vol. 5, 2005.
[2] Hendryx, Stan "Model-Driven Architecture and the Semantics of Business Vocabulary and Business Rules", Hendryx & Associates, 2005
[3] Object Management Group, "Semantics of Business Vocabulary and Rules Interim Specification," 2006, Online at: www.omg.org/cgi-bin/doc?dtc/06-03-02. Accessed: 25/10/2007
[4] R. G. Ross, "The Business Rules Manifesto," *Business Rules Group. Version* 2 (2003).
[5] R.T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California - Irvine, 2000.
[6] D. Wu, B. Parsia, E. Sirin, J. Hendler, D. Nau, "Automating DAML-S Web Services Composition Using SHOP2," Proceedings of the Second International Semantic Web Conference (ISWC2003), 2003.
[7] L. Richardson and S. Ruby, "RESTful Web Services," O'Reilly Media, Inc., 2007.
[8] J. Yang, M.P. Papazoglou, and W.J. van den Heuvel, "Tackling the challenges of service composition in e-marketplaces," Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, 2002. RIDE-2EC 2002. Proceedings. Twelfth International Workshop on, pp. 125-133, 2002.
[9] M.J. Hadley, "Web Application Description Language (WADL),"
[10] C.J. Date, "What Not How: The Business Rules Approach to Application Development," Addison-Wesley Professional, 2000.
[11] D. Obasanjo , "Google Base Data API vs. Astoria: Two Approaches to SQL-like Queries in a RESTful Protocol," 2007, Online at: http://www.25hoursaday.com/weblog/2007/07/13/GoogleBaseDataAPIVsAstoriaTwoApproachesToSQLlikeQueriesInARESTfulProtocol.aspx. Accessed: 25/10/2007

[12] P. Castro, “Overview: Microsoft Codename Astoria,” Microsoft Corporation, 2007, Online at: http://astoria.mslivelabs.com/Overview.doc, Accessed: 25/10/2007.
[13] Google, "Google Data APIs (Beta) Developer's Guide," Online at: http://code.google.com/apis/gdata/index.html, Accessed: 25/10/2007.
[14] Microsoft Corporation, “Web Structured, Schema’d & Searchable (Web3S)”, 2007, Online at: http://dev.live.com/livedata/web3s.pdf, Accessed: 25/10/2007.
[15] J. Gregorio, B. de Hora, “The Atom Publishing Protocol,” The Internet Engineering Task Force, 2007, Online at: http://tools.ietf.org/html/rfc5023, Accessed: 25/10/2007
[16] Amazon, “Amazon Simple Storage Service (Amazon S3),” 2007, Online at: http://www.amazon.com/gp/browse.html?node=16427261, Accessed: 25/10/2007
[17] Y. Goland, E. Whitehead, A. Faizi, D. Jensen., “HTTP Extensions for Distributed Authoring--WEBDAV,” Microsoft, UC Irvine, Netscape, Novell. Internet Proposed Standard Request for Comments (RFC), vol. 2518, 1999.
[18] R.Fielding, J.Gettys, J.Mogul, H.Frystyk, T. Berners-Lee., “Hypertext Transfer Protocol--HTTP/1.1. RFC 2616,” The Internet Engineering Task Force, 1999.
[19] R. Chinnici, J. Moreau, C. Ryman, S. Weerawarana, “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language,” W3C Working Draft, vol. 26, 2004
[20] Ion, M., Telesca, L., McGibney, J. and Botvich, D. OPAALS Deliverable D4.3: “Trust Model for the DE”, 2008.

5. The Agent-based Digital Business Ecosystem network (UniS: A. Razavi, S. Moschoyiannis, P. Krause)

The Web has yet to realise its full potential. In this chapter we focus on work to enable small to medium enterprises (SMEs) to engage in collaborative e-commerce, in a way that maintains an open competitive environment free of (the current) dominance by large organisations. As with Chapter 3, we repeat in condensed form, some material from D3.2 in this discussion in order to provide a self-contained discussion about the current state of the architecture.

In achieving this goal, we see a convergence between our work and the goals of the Semantic Web. We argue that our work is foundational to realising the potential of adding semantic content to Web 2.0. In section 2 of this paper, we establish the connection between our existing research and what we see as the foundational components of the Semantic Web. The main body of the paper elaborates our framework for supporting Digital Business Ecosystems of SMEs. Finally, we return to summarise how our work contributes to enriching the future of the Web.

The current work by UniS, in contrast with conventional Semantic web literature, does not talk about derived technologies, potential progresses or discussions about microformats, natural language search, data-mining, machine learning, and artificial intelligence technologies. Instead, our approach focuses on the enormous potential improvements in DEs by applying virtualization, and more importantly tries to point out a more controversial subject which is questioning the feasibility of the Semantic web. We show the feasibility of the concept of the Semantic web, in a business environment (DE), by using an agent-based approach.

Our virtualizations in the network, offer specific abilities for this software agent model, not only in terms of modelling its business activities through transactions, but also efficiently connecting to any other user (or software agent).

5.1- The content, links, and transactions

According to W3C, the Semantic Web is an evolving extension of the World Wide Web in which web content can be expressed not only in natural language, but also in a format that can be read and used by software agents, thus permitting them to find, share and integrate information more easily [20]. Perhaps more precisely, W3C director Sir Tim Berners-Lee originally expressed the vision of the semantic web as follows:

"*I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the **content**, **links**, and **transactions** between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize*." [ref: Berners-Lee, Tim; Fischetti, Mark (1999). Weaving the Web. Harper San Francisco, chapter 12. ISBN 9780062515872- ref: 24]

There are three important components in W3C's definition; content, links and transactions. First, we clarify these components in a Digital Business Ecosystem and then the design and complexities can be discussed. A Digital Business Ecosystem, as a service-oriented environment, should enable businesses (SMEs) to engage in distributed business transactions [4]. The structural atomic components of each business transactions are web services, which are described by some description language readable by computer programs as well as (sometimes less easily) by humans (description languages such as WSDL, SDL).

As we are involving with service providers in the deployment level [12], the content will be web-services and especially their descriptions on one hand and any composition of them as "on-the-fly-services" [11] on the other hand. The links provide access to/from service providers to consumers and/or other business to execute a transaction. Transactions are business activities between businesses (and consumers).

The complexity is that a Digital Business Ecosystem typically:

- may not be a fully connected network.
- has a very dynamic nature. Especially because of the significant involvement of SMEs, this dynamicity will be in all levels of the semantic web: the nature of web services can change (content); the relationships between participants may change (links); and the nature and quantity of business activities between participants can change too.
- does not have potential for creating all sort of links between different content or service providers (this is one of the side-effects of it not being a connected network,).
- is relying on centralised controls for business activities (frameworks such as ws-businessactitivities, BTP and using ws –coordination framework).
- is not fully resistant against failures and errors.
- is not stable, with traffic bottlenecks and other uncontrolled parameters easily affecting the environment.

5.1.1. First Step

As a first step, we purpose a business network to enable networked organisations to engage in distributed business transactions [4] that realise their core business activities. If such a network is to support B2B interactions between different businesses it should be fully distributed (no central point of control for transaction or network operations), should also offer a consistent model for performing transactions and theoretically and practically each node should be able to have a link to the other node. This means it should be highly resistant to fragmentation.

In addition, given the nature of the internet, there is always the possibility of failure at the transaction level, which should be recoverable and such a procedure must be supported and assisted by the underlying network. The ability for choosing alternative paths/scenarios of execution is another important issue on the transactional level, as well as requirements for feasibility to create links between any two participants.

To reach such a goal, in our earlier approaches we introduced a fully distributed transaction model [1],[4] and localised coordination framework [3]. Furthermore by introducing distributed recovery management and concurrency control, the system's resistance against failure has been increased and a self recovering model has been designed [13]. To integrate the model into such a network, which has high connectivity, and to provide a dynamic topology, we introduce an agent based model. Each local agent is responsible for integrity and consistency of its data and local state. These software agents together provide virtual levels which offer a self-organised network, which reacts against changes during time and is able to change itself for adaptation or recovering against failures.

5.1.2. Software agents on behalf of people (SMEs)

In general, the Digital Ecosystem's software agents need to be able to:

- Gather and store local knowledge (local contents);
- Collect and accumulate knowledge from outside (external links);
- Manage content;
- Promoting the stored contents for outside and provide external links;
- Process business activities and transactions.

As we are working in a service-oriented business environment, the contents are services (their descriptions) and the structural descriptions of on-fly-services (composite services). Meanwhile we have not considered any limitations for using a customized XML or mixture of OWL, RDF and some derived XML from a XML schema for contents (we may address these issues in our later works).

5.2- Structure of a DE's Software agent

Figure 1, shows an overview of the local agent structure. Such a system includes a Local Web Services Informer, a Local Service Repository, a Web Service Information Investor, a Global Service Repository, a Web Services Promoter and a Local Coordinator. We describe each entity in more detail in the following;

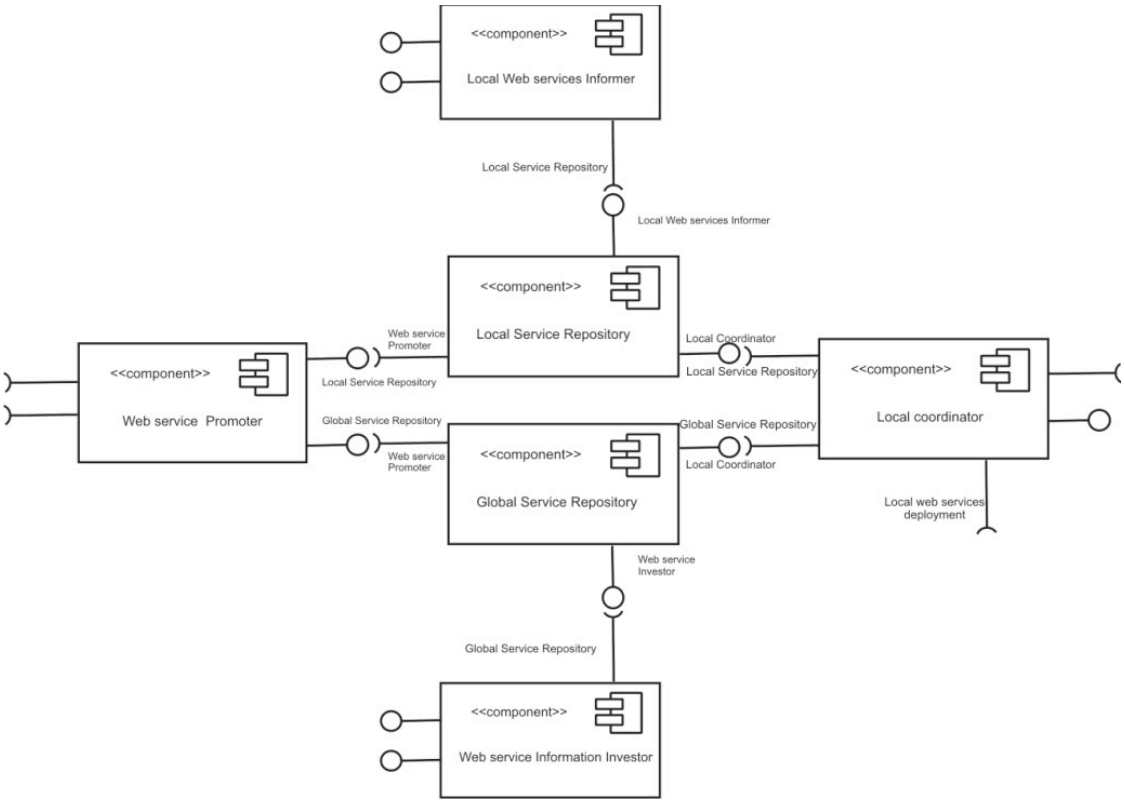


Figure 1. DE Software agent structure

5.2.1. Local Web Services Informer

A Digital Ecosystem is supposed to support any type of web service, with any protocol in a loosely-coupled manner (local autonomy for SMEs). In addition, it should support a proper commit protocol for long-running transactions (business activities as discussed in [1],[3],[4]).

In order to provide the Initiator of a transaction with the basic view about the web service which is to be to be used on its transaction, as well as the limitations / restrictions of the particular web service, we need to gather information about the web service. This information can be provided by each web service after its creation in some description language such as WSDL and/or can be provided manually by the service provider (SME or any business) that provides the web service.

Furthermore, service providers may regularly change their web service protocol, parameters, etc. (in the case of SMEs, this is highly expected); therefore the possibility for updating this information is necessary too. As a result, we need to provide two interfaces for keeping this information in the local agent, as the component also requires an interface to the local repository.

5.2.2. Local Service Repository

The Local Service Repository keeps information about each local web services in the platform (SME). This information is some description of each web service (for example it can be in SDL or WSDL) and any extra information such as availability, last updates and so forth (which may help other SMEs to have a clearer picture of that particular web service), can be included too. In the first place (as a component-based approach), the Local Service Repository should provide an interface to the Local Web Services Informer, e.g. for accessing the local web service description records. The next interface provided by the Local Service Repository gives access to the Local Coordinator to use the web service descriptions for creating and running a transaction.

Any updates, modifications or even the creation of web services should be promoted (at least for other partners with whom they are collaborating in running a transaction). That's the main reason the Local Service Repository requires an interface to another component, namely the Web Service Promoter, whose purpose is to promote the web services to other agents.

5.2.3. Web Service Information Investor

The structure of the local agent we considered in Figure 1 looks symmetric for both the local and the global view of web services. Therefore the Web Service Information Investor, as a symmetric component for Web Service Informer, does a similar job but this time for global web services.

It provides two interfaces for creating a new web services record and updating the current web services. In addition, it requires an interface to the Global Service Repository (the symmetric component to the Local Service Repository).

5.2.4. Global Service Repository

Similar to the Local Service Repository, the Global Service Repository provides two interfaces: one for the Local Coordinator to access the web services record descriptions and the other one for the web services Investor to make changes on the Global Service Repository. We will see later (Section 5-4), that the Global Service Repository plays a key role in persisting and migrating knowledge about the resources that are available in the Ecosystem.

The first interface plays a critical role for the Local Coordinator in making the decision about the protocol and the method for applying it on the transaction model. At the other side of the local agent is another participant (SMEs) which may change its web services descriptions regularly and even service availability can be an issue too. The second interface is important too, as updating the Global Service Repository is crucial.

The Global Service Repository should be able to inform the Web service promoter, as soon as any changes occur for its records. That is why it requires an interface to the Web service Promoter.

5.2.5. Web Services Promoter

The Web service Promoter is an important part of the local agent, as it reflects the situation of the web services of a local agent and the web services of any other connected agents to that particular agent. This can be done by using two interfaces, which are provided for the Local Service Repository and Global Service Repository respectively. Meanwhile the Web Service Promoter requires two interfaces from the other agent to be informed of the latest situation of its local web services and any other web services, which are communicating with it.

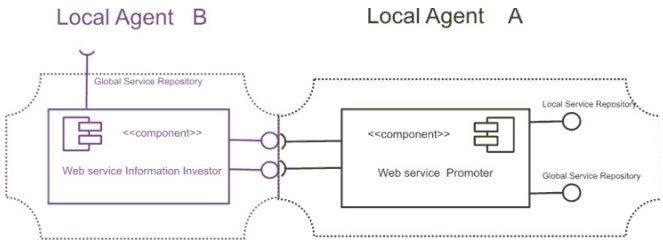


Figure 2. Web service Promoter connections

In fact two interfaces for this component should be provided by the Web Service Information Investor of the other agent. Figure 2 shows this connection between Local agent A and Local agent B. When any changes happen for some records of the local or the Global Service Repository, they use the Web service Promoter's interfaces. The Web service Promoter in turn can use the interfaces provided by the Web Service Information Investor in Local agent B, and the Web Service Information Investor at agent B can update its Global Service Repository if needed (because in some cases it could be done already). As a result, the Global Service Repository of agent B will use the same interfaces for the Web service Promoter at agent B and this will be done for any connected agent to Local agent B. In this way, any changes on connected agents can be updated quickly.

5.2.6. Local Coordinator

The kernel of the local agent is the Local Coordinator. Other components provide information for a Local Coordinator (on the local machine or even for a remote agent). The Local Coordinator facilitates our transaction model to be applied for complicated business activities (long-running transactions) as well as simple transactions.

Generally the Local Coordinator requires an interface from the Local Service Repository for gathering the information about local web services which enables it to provide the preparation and commit phase in a two phases commit (2PC) protocol. This normally can be handled by a transaction content in response to a transaction request (Script).

For communicating with another agent (its Local Coordinator), the Local Coordinator as well as *providing* an interface, *requires* an interface from the remote agent too. The Local Coordinator also requires an interface from the Global Service Repository, especially when it acts as an Initiator of the transaction. This makes it possible to create the transaction script based on the knowledge of the other agents' web services. Ultimately, it requires the interface from its local web services to able to invoke them.

5.3- Digital Business Ecosystem Network

Our previous work [3] has described a distributed model of multi-service long-running transactions which has been designed for open collaborations within a community of businesses (SMEs) in digital

ecosystems. The transaction model provides the capability for efficient recovery management, in terms of preserving as much progress-to-date as possible (omitted results), and provision for alternative scenarios or paths of execution (forward recovery). Using local coordination not only avoids any violation of local autonomy but also provide a fully distributed model for the transactions to be executed.

We now look at the durability and reusability of the transaction and stability of it (as the dynamic nature of SMEs) as areas for improving the expected quality of a DE. It is important to avoid the abortion of the transaction even when some (or even all) participants are temporarily disconnected. This problem has been solved when one of the participants (at each nested part of the transaction) is/are disconnected. But we try to provide a highly reliable environment which can cope with dynamicity of SMEs and high probability for their disconnections.

It is very important to keep the result of a successful or unsuccessful transaction (even by considering regular unavailability of initiator and other participants). Another issue is lookup algorithm. Meanwhile we might provide even more knowledge about the unavailability of SMEs (or their services) based on their natural behaviour. This can be argued by the probability of fragmentation on such a network (ref: D3.1, D3.2). Our goal is not just providing a network structure for answering these requirements but also reusing fragmented network structures provided by the transactions to create a fully connected network.

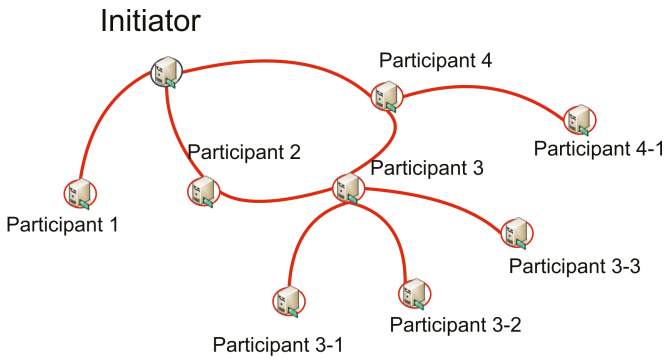


Figure 3. Distributed coordination for executing transaction

Figure 3, the actual execution of a transaction is in a fully distributed manner with each participant using its local coordinator. This creates a temporary network between service providers which will disappear after the transaction is finished. However, this network has some unique characteristics which are worth keeping for later use. This temporary business network inherits all of the transaction model properties (ref: D3.1, D3.2); in one hand, it is loosely-coupled which gives full local autonomy to platforms, and on the other hand it has resistance to failure and can be recovered - it can even handle short-term disconnections as the traffic is not focused on a centralised point. Meanwhile the platforms involved in a transaction are in a related domain and there is probability for them to do similar transactions again.

5.4- Virtual Private Transaction Networks

We call the temporary network created by a 'Transaction' a "Virtual Private Transaction Network" (VPTN), as apart from the participants in the transaction, naturally they are shared with other platforms and normally they are not created as an actual network. Because of the specific properties of these networks we consider a component for keeping them and adding the ability for re-using them.

Figure 4 shows a collection of these networks. Clearly, VPTNs are a collection of fragmented networks, apart from occasional overlaps between different transactions which is also transparent.

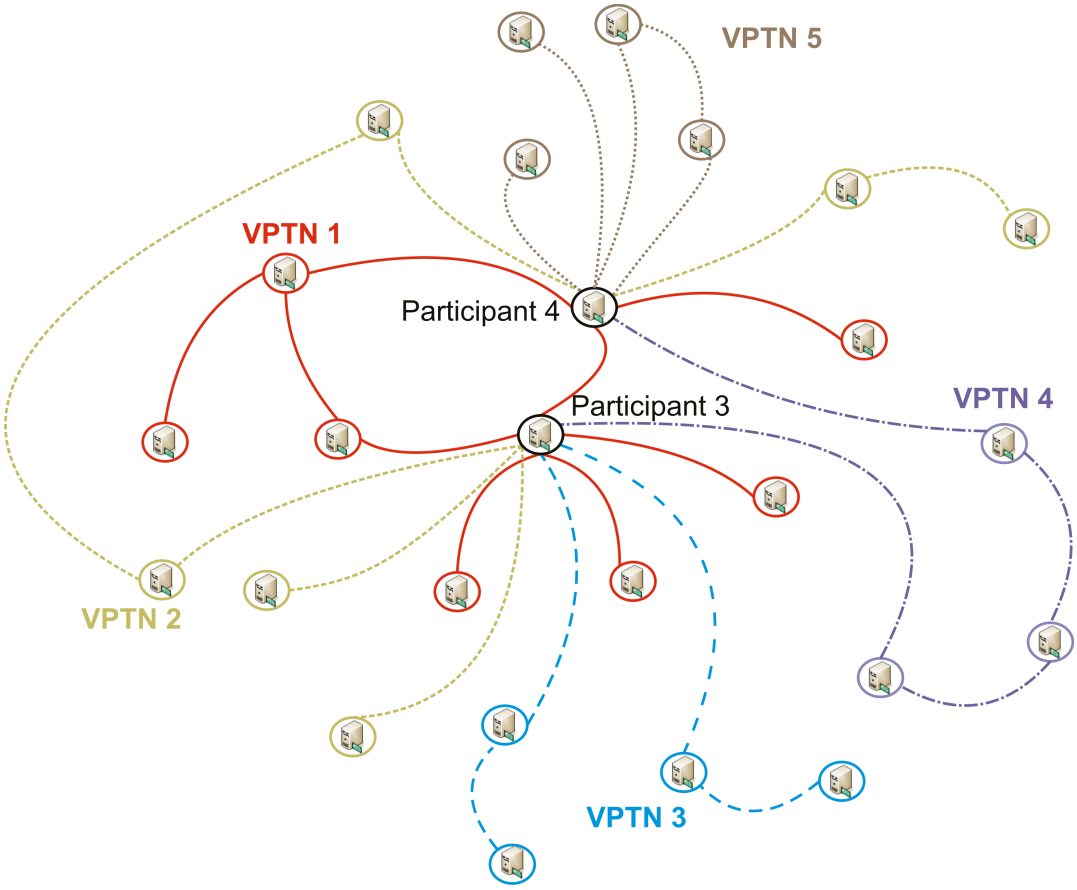


Figure 4. VPTNs

The Global Service Repository's (introduce in III.B) main duty is to keep our virtualization permanent according to the life-time of each business agent. Therefore these VPTNs will be saved by it. In the aspect of implementation, the simplest option which we considered is using a database but we consider the adoption of this to other forms (such as some standard schema which may show the connection and relationships and reflect the semantic of the association in a standard way). An example is given in Figure 5; the Local Coordinators are connected to the Global Service Repository, they can update links and information about other participants and their services can be saved. In this way, VPTNs can be saved permanently.

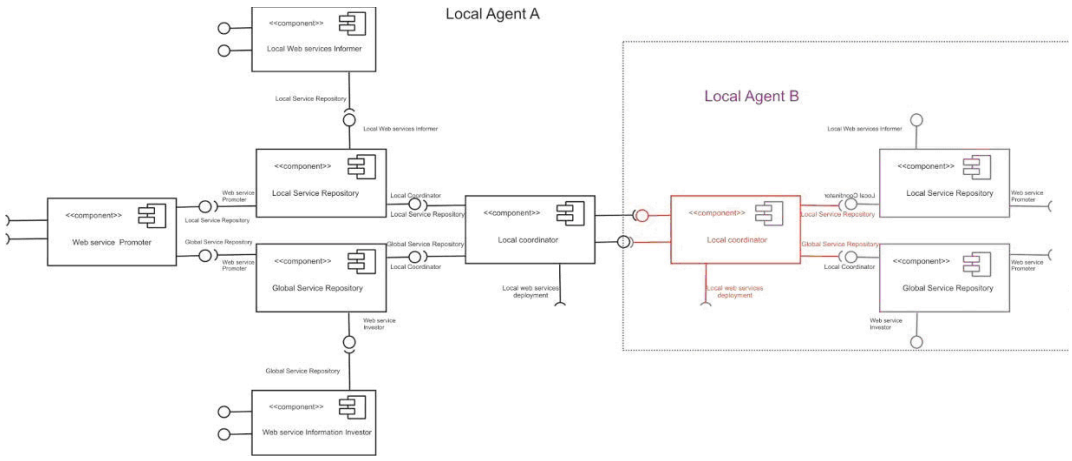


Figure 5. Global Service Repository connections

It is important to mention, keeping this information is not the only duty of 'Global Service Repository'. Actually the 'Global Service Repository, by using a 'Web Service Information Investor', tries to update its

information about other platforms (web services) based on the architectural requirements (which may be clearer by the end of this paper) and at the same time, by using 'Web service Promoter', tries to transfer its records to the other platforms for creating a more stable network (which will be explain in the rest of this chapter). Figure 5 can explain these connections between 'Global Service Repository' and 'Web service Promoters' and 'Web Service Information Investor'. (there are more here in connection with semantic web, which will be added).

5.4.1. A measurement for Platform stability

In order to increasing the lifetime of each VPTN and avoid business transaction abortion during temporary disconnection of some participant in a VPTN, we need to increase the stability of VPTNs over time. We try to find some measurement of stability for each node in the network. It turns out that the probability for availability of each node is the important factor. For this reason, in this section we try to analyse this factor by using service availability and later on, we can use this to ensure better stability in the network. Our purpose is to find a good candidate in each VPTN for keeping information about the private network in its repository and in this way have maximum stability of the network in time (for example, it may be able to keep this information for some hours). Then in the next part we try to improve this network towards a fully permanent network.

It would be unreasonable (and not feasible) to expect nodes to be online all the time and thus stability is determined on the basis of declared availability.

For finding a more precise and computable measurement for node stability, first we introduce an important property for each node, called Expected Availability Time (EAT). This is the time when the node is expected to be available and online in the network (Figure 6 shows an example of EAT for a node in the network). The node stability is then calculated as the actual availability of the node against this expected time. These are typically different, since during its EAT the node may be disconnected. These disconnections will reduce stability (reliability) of the corresponding node in the final selection. This notion of stability can be simply calculated as below:

$$NodeStability = \frac{EAT - DisconnectionPeriod}{EAT}$$

It can be seen that the closer NodeStability tends to 1 for a node, the more stable the node is (which can be understood as more reliable or predictable). For calculating the stability function of a node, in the first instance we use its participants in a transaction (other nodes in the same VPTN) to check its availability behaviour. At the moment, we have considered EAT as a part of service provider's (SMEs) business model which is provided by each of them when they join to the network (and they may change it, if the nature of their business changes). It should be noted that other approaches can be considered for calculating the EAT - for example, it is possible to use an algorithm based on the network neighbourhoods for calculating EAT which would allow it to vary over time.

Further refinements to this definition of node stability are in hand. For example, node stability may also have a seasonal effect, and indeed additional service-level quality factors may usefully the inform choices of nodes to include into VPTNs.

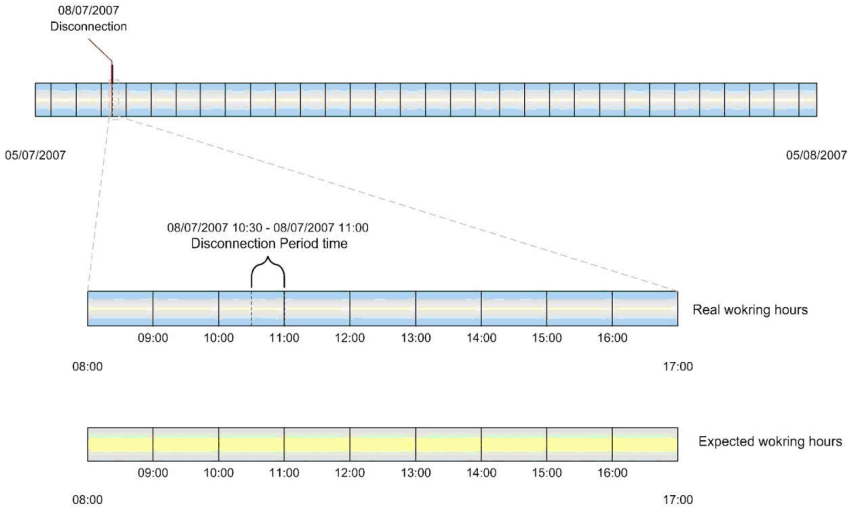


Figure 6. Expected Availability Time

By finding the most stable node in each VPTN, we can rely on that node for keeping uncommitted transaction information or even saving the commit transaction's results for providing better global durability and reusability. But still one may ask, what will happen if all nodes in a VPTN (including the most stable one) are disconnected together?

5.4.2. Virtual Service Network

Apart of the unanswered question about a total disconnection in a VPTN, the virtualization in the transactional level provides us with a fragmented network but still does not fulfil the semantic web's link feasibility from any node to the other. For answering this requirement, we provide another virtual level, which offers a connected network between services (service providers). Figure 7 shows our different level of virtualizations.

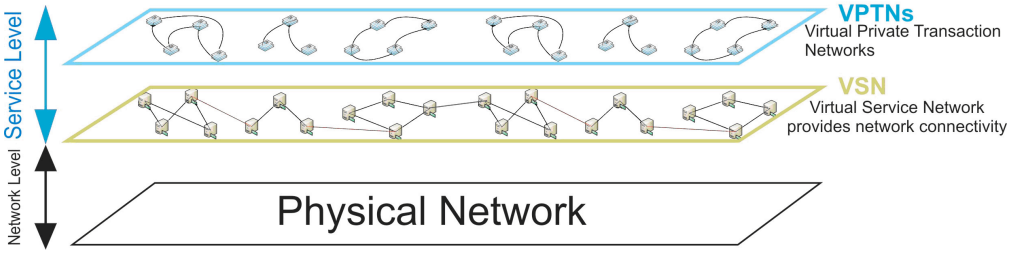


Figure 7. Virtualization in Digital Business Ecosystems

An important question will be: how can this network be created without external controls and how can software agents manage it based on the dynamic character of a Digital Business Ecosystem?

5.5- Increasing connectivity in a dynamic environment

Clearly the simplest way for provide a fully connected network is by connecting VPTNs together. The best candidates for connecting VPTNs together are the most stable nodes in each VPTN. Figure 8 shows a demonstration of this connected network. In this paper, we mostly focus on the result of such a network (which shows the characteristics and necessities of it) and do not focus on the birth model.

By connecting VPTNs, we mean the 'Global Service Repository' of each candidate in each VPTN will be connected to the candidate of the other VPTN. In this way we can improve the stability of the connected

network (the maximum time for the network to be alive). However, we cannot warranty full stability of the network and still cannot avoid the occasional fragmentation (because even in the best case, it is dependent on each platform's availability and if the total online time of all stable nodes cannot cover 24 hours, our network will collapse for some period of time, precisely that in which all of them are not available).

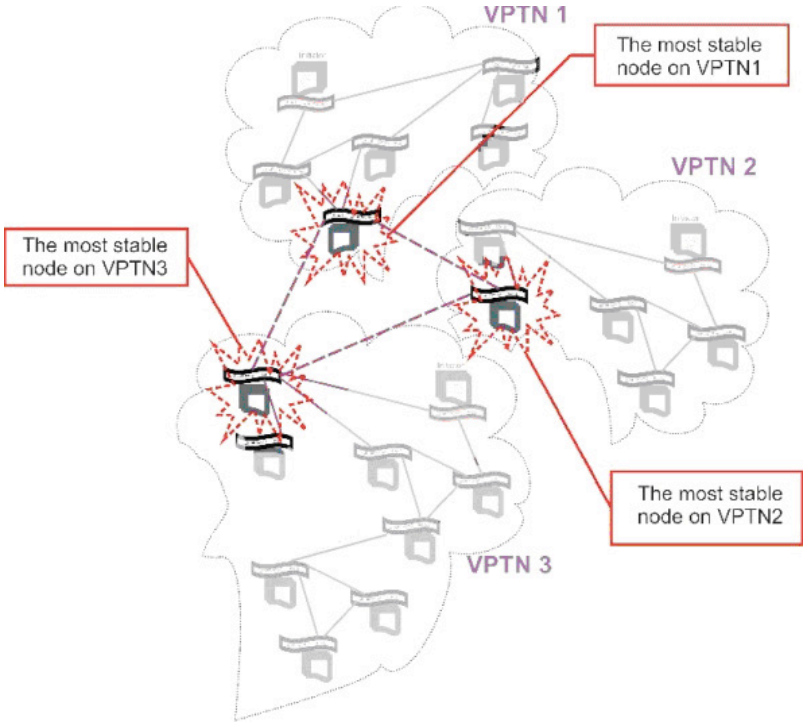


Figure 8. Connecting VPTNs

5.5.1. Super peers and permanent nodes

The conventional solution to the problem of fragmentation, which has been used by several P2P networks ([9], [10]), consists of introducing an extra layer to the network; the so-called super peers. Actually the super peers are decentralised servers, which are intended to provide reasonable connectivity and avoid the fragmentation in the network. Depending on the size of the network, the protocol used and the number of super peers, each super peer manages a number of nodes and can check their availability. At the same time, each super peer provides a strong link to the other super peers and in this way the design ensures that there is low probability for fragmentation.

The primary necessity for having super peers is providing stable nodes which are online all of the time. This means super peers are expensive nodes with costly maintenance requirements. It should also be noted that their resources are used for facilitating network operation management tasks. When considering such a solution for a digital ecosystem environment involving SMEs, the question arises as to who is going to provide such nodes? Apart from feasibility issues, most SME business models militate against this.

Additionally, during peak time the pressure of high traffic can result in a bottleneck on super peer nodes and because of the connectivity role of super peers, the whole VSNs and consequently VPTNs (and the corresponding business activities of SMEs) will come under serious risk. It could be argued that the problem may be countered by providing maximum facilities and additional resources on super peers, but this may address the problem only temporarily. Powerful super peers will still need to be online and monitor the whole network at all times, processing redundant data and producing overheads waste at off-

peak times of the network while they will be continuously under pressure at peak time while the network grows (more nodes join).

Moreover, and even if it were possible to find suitable SMEs willing to provide permanent nodes as super peers, these may change their business model and after some time may not find it useful to provide a permanent (and expensive) node anymore. It is not advisable and may not even be possible to force small-to-medium enterprises to be constrained into a static business model and stable behaviour for the sake of stability of the DE infrastructure.

Perhaps even more importantly, the super peers solution results in a static topology for the network as these nodes are pre-selected and their role is pre-determined in the network. This is by no means satisfactory in a highly dynamic environment of a digital ecosystem where the idea is that the network topology changes continuously to adapt to its very usage and demands of the participating entities. The evolving nature of the DE is intended to reflect the congestion of network packages and nodes that change from time to time.

It transpires that dealing with change and adapting to ever changing requirements is unavoidable in the content of a digital business ecosystem. This leads to thinking about a design solution that provides a dynamic topology that continuously evolves to echo changes in the participating entities or nodes. Our approach to the business network design is based on clusters of nodes for providing permanent clusters, rather than permanent nodes as is the case with super peers, and is described in the following sections.

5.5.2. Permanent Cluster

As mentioned before, in contrast with conventional super peers, we try in our network design to move towards a more dynamic architecture which does not rely on just a few permanent nodes. Central to our approach is finding permanent clusters on the network. More specifically, we are identifying aggregations of stable nodes, where node stability is determined as in the previous section. For doing so, the most stable nodes from different time zones must be chosen, in a way that they cover 24 hours. In fact, we are trying to find permanent clusters through the most stable nodes.

The important part in determining permanent clusters is discovering different aggregations of these time zones which can cover 24 hour availability. Any union of the stable nodes in the aggregations (which provides 24 hour availability) are actual permanent clusters. Figure 9, shows the simple situation in which the most stable nodes have been selected from two sets of time zones which can cover 24 hour service availability to form permanent clusters.

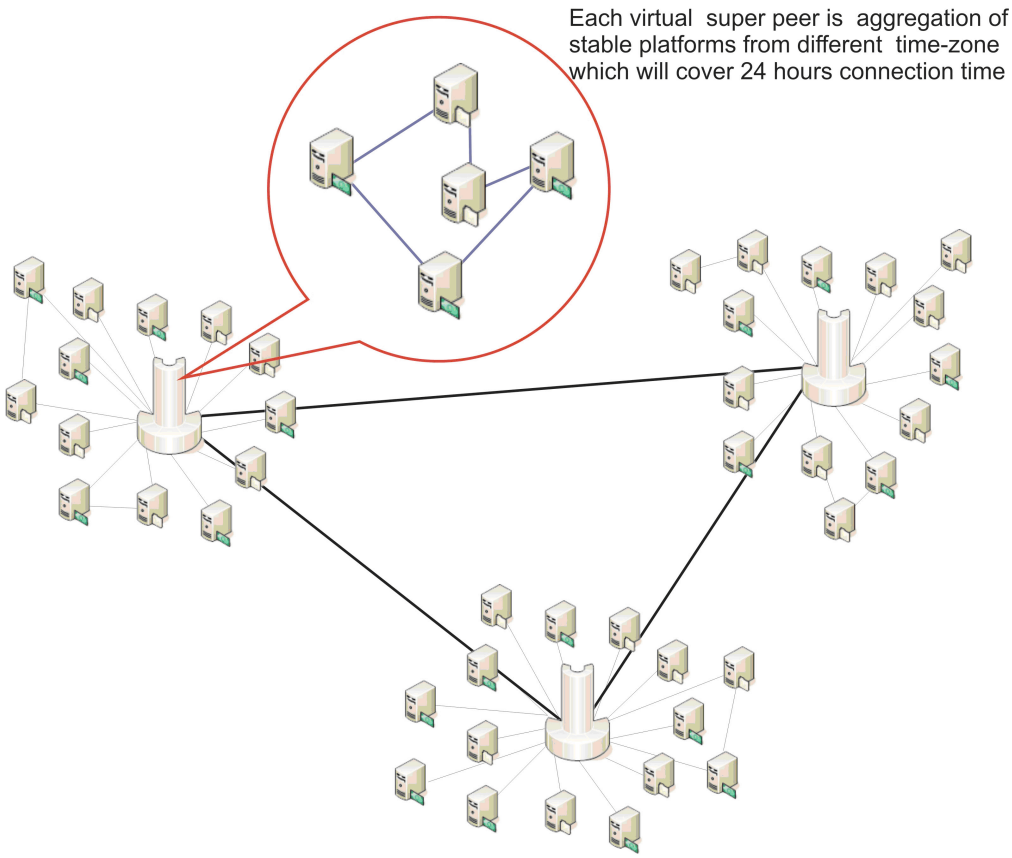


Figure 9. Permanent Clusters and Virtual Super Peers on DE

5.5.3. Virtual Super Peers

By using stable nodes from permanent clusters, as is shown in Figure 9, we can create Virtual Super Peers (VSPs) which are effectively permanent clusters of nodes in the network. These can provide the desired stability for the network. The strong connection between the virtual super peers themselves on the one hand and the connection between them and their nodes on the other, decreases the probability for fragmentation. Depending on the level of reliability required for the network, it is possible to include further redundant stable platforms from each available time zone. For example, in Figure 9 we have included two stable nodes from one time-zone and three stable nodes from the other one (the green and creamy signs show different time zones).

In this manner, the good connectivity can cause more reliable transactions at the VPTN level. Meanwhile the traffic is spread over the virtual super peers and there is less risk of bottlenecks at peak times. Nodes within a virtual super peer need to keep information only about nodes in their cluster and about neighbouring VSPs so at off-peak time the amount of redundant information processing is reduced dramatically as compared to the classical super peers solution.

Since choosing stable nodes is a dynamic process (it is done based on the stability function, EAT to Disconnection period of a node during EAT, whose value varies over time) the virtual super peers are also formed dynamically. This means the topology can change from time to time and new nodes can be added to the permanent clusters as the structure of virtual super peers changes. A node can become part of a virtual super peer, when its node stability increases and overcomes some threshold. In contrast, nodes that are super peers may not be able to cope with the increased number of connections they get, and possibly increased number of transactions they perform, and so lose their virtual peer status. Within a digital ecosystem for business, SMEs would be expected to invest at that time (in hardware, processing power,

band-width etc.) and become again part of a virtual super peer in future. It is in this sense that the topology evolves to reflect the usage and demands of the participants who benefit from and contribute to the 'sustainability' of the network.

Additionally, network congestion can change the maxi-mum level of node stability (Section…) which in turn affects the selection of the most stable nodes in forming the permanent clusters. High congestion of packages can increase or decrease network reliability (higher traffic on few virtual super peers can potentially create a bottleneck and even cause fragmentation). In a digital business ecosystem, the best part of the traffic is the result of business activities which are effectively long-lived transactions. These have been virtualised in VPTNs and therefore, using the effect of VPTNs for making VSPs and their client nodes, can increase the stability of each virtual super peer.

Furthermore, we expect a reasonable cluster coefficient on the account of having VPTNs as the main building blocks, which we have seen are formed from a transaction. This means its nodes are in relevant domains – by connecting them to several VSPs we actually increase the probability for that. We also expect a fair distribution degree on the account of propagating links to VSPs. This means that instead of being concerned with individual links for each node, aggregate links of VSPs come into play.

Finally, reusing business activity results (or service-on-fly as result of composite services [11]) and explorative service composition [12] are other factors which can be considered for higher performance within a digital business ecosystem and can provide potential for creating so-called virtual vendors.

5.5.4. Parametric algorithm for choosing VSPs

In the first step, the most stable node in each VPTN (participants of a transaction) should be selected for keeping vital information about the transaction and its participants. In this sense, the network provides a level of durability without any extra cost from participants and it covers omitted results, a problem relating to preserving as much progress-to-date as possible in the event of aborting a transaction. An extended lock mechanism for recovery management in long-lived business transactions has been described in [13] which contains further details on omitted results as well as other aspects of compensation in long-running transactions.

The best candidates for connecting VPTNs together are the most stable nodes in each VPTN. Figure 9 shows the internal structure of each VPTN and the connection between VPTNs. The internal structure of VPTN contains a lot of information from the transaction level such as log structures, lock schemes for ensuring consistency in recovery mentioned above, local coordinator design, formal analysis of the required interactions and compensations, along with alternative scenarios for forward recovery.

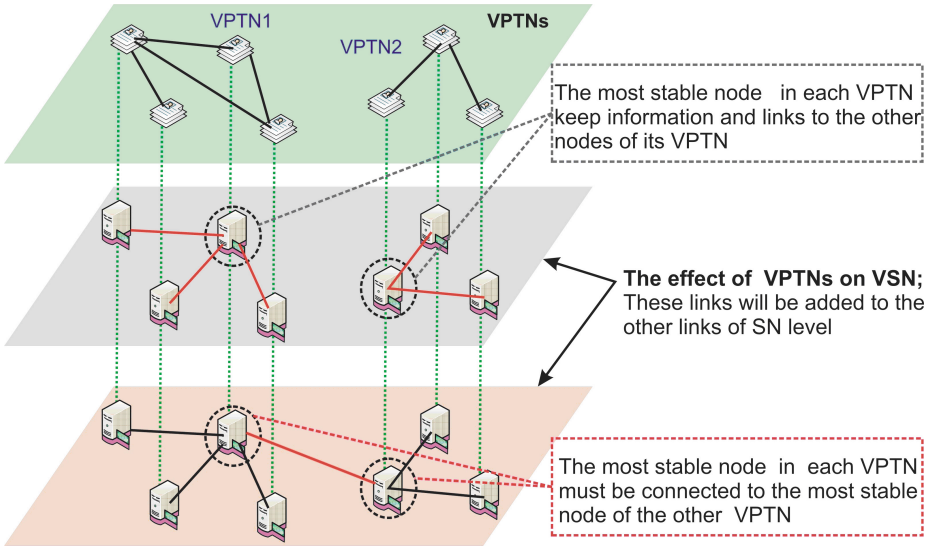


Figure 9. Optimisation from VPTNs to VSN

The direct effect of connecting VPTNs together is to raise the cluster coefficient of the network. Conversely, connecting the most stable nodes of VPTNs together provides the opportunity of choosing the best candidate locally between these stable nodes for the permanent cluster. Choosing nodes of the permanent cluster in this way, results in a virtual super peer that provides fair traffic distribution at the VSN level (each virtual super peer will take care of its local VPTNs). The main concept behind forming permanent clusters stays the same, i.e., selecting the most stable nodes from different time zones which can cover 24 hours online time.

5.6- The network in Practice

The important question is about feasibility of applying basics of the semantic by using these virtualizations. On the other hand, we want to show, it is possible for any agent to not only deploys transactions in a collaborative way, but also traverse the virtual network without any fragmentations. For doing so, our transaction schema is a heterogeneous model [ref: D3.1, D3.2, ieee paper] which is able to work with different standard and protocol. On the other hand, we considered a general structure for our local agent [D3.2 and recall III] to provide ability for working with XML, XML Schema, RDF, RDF Schema and OWL.

However, the most important question will be about the practical stability of the network. As the most stable node in each VPTN is the best candidate for keeping the transaction information, the corresponding business activities will have increased levels of reliability. The fact VPTNs are used initially in the design of the business network, and are connected through their most stable nodes which are determined dynamically, allows in most cases the candidate platform to avoid the full rollback or compensation of the transaction when some participants of the long-lived transaction get disconnected within its du-ration. This can be considered directly in the design of the recovery mechanism for such a transaction model, as done for example in [13] for the distributed transaction model de-scribed in [3]. Another expectation of the network design we have proposed is that the dynamic topology resulting from the selection of virtual super peers, which relies on the stability measurement of each individual platform in each VPTN, reduces the probability of fragmentation. Certain evolutionary models studied in biology exhibit some characteristics of this network design. Meanwhile some practical simulations can compare the theoretical behaviour and practical status of the network in different situations. These aspects are discussed in further detail next.

5.6.1. The Network in Practice

We have seen that we are dealing with a highly dynamic environment where there is no central point of control and a high probability for failure (in a transaction or the network itself). In the design of a business network for this environment we have considered a dynamic, ever changing topology. It would however be desirable to be able to somehow guide the way this topology evolves. Considering the requirements of DE for business, we propose to draw upon the evolutionary growth of metabolic (signal transduction) networks, as studied in the work of Rzhetsky and Gomez (e.g. [14], [15]) in designing the birth and growth model for an autopoietic P2P network to sup-port long-running business transactions in the OPAALS project (see [1], pp. 77-94). It turns out that the evolutionary growth in molecular networks exhibits scale-free characteristics while it also has some interesting properties with respect to network connectivity. More specifically, the frequency of vertices connected to exactly k other vertices in metabolic (signal transduction) networks follows a power-law distribution. The distribution function degree is equivalent to:

$$P(k) = c.k^{-\gamma}$$

where c is a normalising constant and diverges across networks (but usually has a value between 1-3; in our simulation was 2.34) and the network follows the classical Barabasi-Albert model [16]. This growth model says that the total number of network vertices is more than three times the number of nodes, which shows the connectivity (even with-out using VSPs) to be quite good. But there were specific weaknesses which do not seem to be solved without introducing the VSPs conceptual model.

5.6.2. Achilles heel of the network

The basic network as a scale-free network follows the power law distribution, which means most nodes will have a few links and a few nodes will have a large number of links (see Figure 10). This may result in a high dependency on a few number of nodes which have a large number of links. Such nodes actually play the role of hubs in a typical scale-free network. Thus, the network would already be vulnerable since any smart attack on hubs (or even a series of accidental failures), may cause fragmentation on the network (creating islands in the network), As a consequence, any running business transactions will be discontinued and a fragmented network can be extremely costly to repair (de-fragmentation) as discussed before.

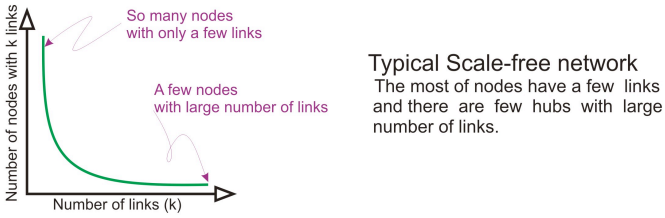
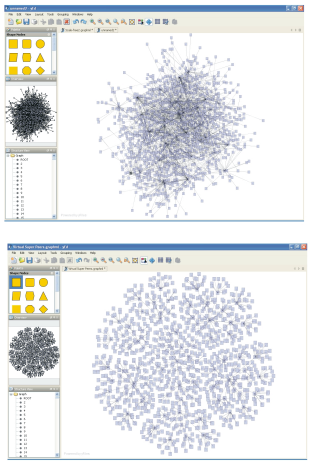


Figure 10. Power Law Distribution

Another problem has to do with the inconsistency of such a model with the dynamicity of a digital ecosystem and the versatility of SMEs business models. Hoping to have stable and permanent hubs to warranty the stability of a few individual nodes is in contrast with the very nature of an ecosystem. Also, SMEs may not provide stable and permanent nodes for hubs at all. Therefore, there is possibility for fragmentation even without any external attack or physical failures. As a result of this, the network may suffer regular transition periods between exhibiting the characteristics of a scale-free network and those of a fully random network (with potential islands).

5.6.3. Agent-based DE network's experience

Our experimentations and results show the proposed model to use the advantages of a conventional scale-free network, but at the same time has built-in capability for coping with events which typical scale-free networks are vulnerable against. This is depicted in Figure 11 (our network is DBE/OPAALS scale-free network).



Conventional Scale-free network
Power Law Distribution degree
Small world properties
Vulnerability for smart attacks

DBE (OPAALS) Scale-free network
Power Law Distribution degree
Small world properties
Resistance against smart attacks

Figure 11 Digital Business Ecosystem network in comparison with a conventional network

The major effect of our proposed model is that the actual hubs are virtual super peers. As mentioned in Section V, VSPs are aggregations of several nodes that create a permanent cluster. The dynamic nature of VSPs makes them adaptable to changes in business models or more generally the versatile nature of SMEs. In addition, the fact 'platform stability' varies over time allows for the formation in virtual super peers to also change (if the corresponding platform stability changes).

5.7- Conclusion and further work

We have provided some level of virtualization which offers the basic characteristics for semantic web. On one hand, distributed transactions have been modeled in a virtual private transaction network (which supply the semantic links between business partners in a business activity at the same time), on the other hand, virtual service network, as a connected network, provide any software agents virtually to have a link to any other service provider. We have introduced this network as a P2P network, based on the notion of virtual super peers that equips the network with ability for resisting against different types of failure, which support our semantic map of a connected network between services which supports transactions and it is readable by computers as well as human.

In contrast with a conventional scale-free network, attacking a few high degree nodes may not destabilize the whole network. This is because in the first step each virtual super peer is made up from several different platforms, from different domains even. At the same time, the dynamic nature of 'permanent clusters' allows for each participant in VSPs to be easily substituted by another stable platform (recall the discussion in Section V).

In addition, platform failures or traffic bottlenecks may not fragment the network or lead it to a transition state between the random graph and its original topology. Actually, even if a node of a permanent cluster fails or experiences high traffic, this will only cause a substitution of the node with another stable node in the closest level. Our actual simulations show the network to follow a fractal model around virtual super peers which can vary depending on the size of the permanent clusters.

The resulting characteristics of this network foster an environment that potentially enhances the ability of SMEs to compete by means of virtual vendors. Smart composition of services and reuse of profitable

(and successful) transaction results can lead to the creation of virtual retailers in place of large enterprise vendors that wish to dominate the infrastructure and rip any potential benefits. Furthermore, the stable network in a collaborative environment can be used in the continuous creation and sharing of knowledge in the form of business models.

5.7.1. Web 2.0

This approach has been inspired by web 2.0 and used its presumptions and views; we assumed the web as a platform, in our model Network effects created by an architecture of participation [see 17], we considered the business processes are modeled by transactions and business models are enabled by content and service syndication, we assumed Software above the level of a single device by the mean of leveraging the power of the "Long Tail" (Businesses with distribution power) [see 18,19].

Web 2.0 also has been described as an environment where users generate and distribute content, often with freedom to share and re-use. One perceived result is a rise in the economic value of the Web and it has been one of our primary aims in designing distributed transaction and providing the reusability for participants in the transactional virtualization (VPTN). But this was the start of our approach and the current experiences and results are inline to moving forward towards next step which it is 'Semantic Web'.

5.7.2. Future of the web and roadmap for our approach

Already discussion about the next generation of Web (Web 3.0) has been started:

"*Web 3.0, a phrase coined by John Markoff of the New York Times in 2006, refers to a supposed third generation of Internet-based services that collectively comprise what might be called 'the intelligent Web' — such as those using semantic web, microformats, natural language search, data-mining, machine learning, recommendation agents, and artificial intelligence technologies — which emphasize machine-facilitated understanding of information in order to provide a more productive and intuitive user experience.*" [20], [21]

Apart of our roadmap towards 'Semantic web', we have started to figure out other expectations of the new web generation in our work; on one hand our approach shows the possible convergence of Service-oriented architecture and the Semantic web (as it is discussed with details in [22]), on the hand, our research is trying to conduct developing software for reasoning, based on description logic (and provide the feasibility for intelligent agents in the future) [23]. Meanwhile our approach has started the necessary design and emergence of different technologies [20] such as Web services interoperability in distributed P2P solution and at the same time considering not only architectural aspects of the Semantic web but also a primary attempt to offer the full ability for processing Semantic Web technologies in different levels of virtualization. We are trying to complete this merge and move towards deeper aspects of the Semantic web.

5.8 References (for Chapter 5)

[1] A. Razavi, S. Moschoyiannis and P. J. Krause "Preliminary Archi-tecture for Autopoietic P2P Network focusing on Hierarchical Super-Peers, Birth and Growth Models." OPAALS project Deliverable D3.2, 2007 - available at: http://files.opaals.org/OPAALS

[2] Z.B. Daho, Simoni, N. "Towards Dynamic Virtual Private Service Networks: Design and Self-Management", 10th IEEE/IFIP Network Operations and Management Symposium, 2006. NOMS 2006.

[3] A. Razavi, S. Moschoyiannis and P. Krause. "A Coordination Model for Distributed Transactions in Digital Business Ecosystems." In Proc. of IEEE Int'l Conf. on Digital Ecosystems and Technologies (IEEE-DEST 2007), IEEE Computer Society, 2007.

[4] A. Razavi, P. J .Krause and S. K. Moschoyiannis. DBE Project De-liverable D24.5: DBE Distributed Transaction Model, 2006.

[5] B. Martini, F. Baroncelli, P. Castoldi, "A novel service oriented framework for automatically switched transport Network", Sympo-sium on Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International. Pub: 15-19 May 2005, pp(s): 295 - 308.

[6] Kristiansen L. et al. "TINA Service Architecture and Specifications", TINA 1.0 DELIVERABLES AND SPECIFICATIONS (available at: http://www.tinac.com/specifications/specifications.htm)

[7] Sahai A. et al. “Automated SLA monitoring for web services”, Pro-ceedings of the 13th IFIP/IEEE International Workshop on Distrib-uted Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications, p.28-41, October 21-23, 2002.

[8] Z. Li, P.Mohapatra, “QRON: QoS-Aware Routing in Overlay Net-works”, Service Overlay Networks in the IEEE Journal on Selected Areas in Communications (2004).

[9] B. Y. Beverly, H. Garcia-Molina, " Designing a super-peer network", Proceedings. 19th International Conference on Data Engineering, 2003. Pub: 5-8 March 2003, pp: 49-60

[10] B. Yang, H. Garcia-Molina, "Improving search in peer-to-peer net-works", Proceedings. 22nd International Distributed Computing Sys-tems, 2002. pp: 5- 14

[11] Yang, Jian; Papazoglou, M P. and Heuvel, W-J van den (2002), “Tackling the Challenges of Service Composition in E-Marketplaces”, Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, 2002. RIDE-2EC 2002 (IEEE Computer society), pp:125-133

[12] Papazoglou M. P., Traverso P., Dustdar S., Leymann F. and Kramer B. J. Service-Oriented Computing Research Roadmap. In Dagstuhl Seminar Proceedings 05462, Service-Oriented Computing (SOC), pp. 1-29, 2006.

[13] A. Razavi, S. Moschoyiannis and P. Krause. Concurrency Control and Recovery Management for Open e-Business Transactions. In Proc. of Communicating Process Architectures (CPA 2007), 2007.

[14] A. Rzhetsky, S. Gomez, "Birth of scale-free molecular networks and the number of distinct DNA and protein domains per genome", Bio-informatics, 17(10):988-996, 2001.

[15] G. Karev, Y. Wolf, A. Rzhetsky, F. Berezovskaya, E. Koonin "Birth and death of protein domains: A simple model of evolution explains power law behavior", BMC Evolutionary Biology, 2:18, 2002

[16] A. L Barabasi, R. Albert “Emergence of Scaling in Random Net-works”. Science 286(5439): 509–512, 1999

[17] Battelle J. and O'Reilly, T. "The State of the Internet Industry", First Web 2.0 Conference, San Francisco, USA, 2007-11-08.

[18] Anderson, C., "The Long Tail", Wired Magazine ,Issue 12.10 - October 2004.

[19] Anderson, C., The Long Tail: Why the Future of Business is Selling Less of More, Hyperion (July 11, 2006)

[20] Spivack, N. "THE THIRD GENERATION WEB IS COMING", LIFEBOAT FOUNDATION SPECIAL REPORT (last access 10/11/07, available at: http://lifeboat.com/ex/web.3.0).

[21] Markoff J., "Entrepreneurs See a Web Guided by Common Sense" , New York Times, November 12, 2006.

[22] Provoost, L. and Bornier, E. "Service-Oriented Architecture and the Semantic Web: A killer combination?", University of Utrecht, February 10, 2006 (avalaible at: http://lee.webcoder.be/papers/sesa.pdf)

[23] Wainewright, P. "What to expect from Web 3.0", ZDNet, November 29, 2005 (last access: 10/11/07 http://blogs.zdnet.com/SAAS/?p=68)

[24] Berners-Lee, Tim and Fischetti, Mark, Weaving the Web. Harper San Francisco, (1999). chapter 12. ISBN 9780062515872

6. A Simulation for DE Transaction model (SUAS)

The conceptualization and ongoing implementation of a network and service infrastructure for SMEs that bases on biologically inspired similes needs thorough simulation and emulation capabilities that accompany that development. This network and service infrastructure acts as a *Digital Ecosystem* as it is mediating optimum services amongst SMEs that are provided from a pool of services within that network. Finding optimum services relies on Genetic Algorithms that guarantee the best fitness available for the respective services.

In this chapter we present an extended variant of the *EvESimulator* simulation framework that is both, able to perform simulations of an evolutionary framework and to work as a visualization tool for the real-world P2P network.

In the following chapter an introduction to the Evolutionary Environment Simulator (EvESim) framework is provided, that is developed within the DBE project and the extensions so far toward a *peripheral and distributed* simulation framework. This simulation framework should act as common framework where parts of the new research outputs can be tested. Social science can provide data about social networks in a region based on surveys for example which can then be used as a basis of simulating advanced P2P algorithms.

Furthermore, we analyse the drawbacks of the DBE version of the EvESim and present a new and distributed design for the simulation framework. In order to show the issues of the redesign and the functionality of the simulation framework, we run the critical mass assessment and clustering simulations of the DBE and additional a simple transactional model on the new framework.

Success criteria of the work described here are mainly 1) the redesign of the EvESim framework in order to run distributed on P2P or application server framework, 2) the identification of the main components needed for this distribution, 3) the implementation of the first services like a simple Genetic Algorithm and a agentpool container and finally 4) the documentation of the challenges and problems arising from a distribution of services on the DBEs servENT P2P infrastructure under a heavy usage like a distributed agent-simulation.

The evolutionary parts of the EvESim so far are on one hand the usage of Genetic Algorithms for the dynamic service search and composition in the model, and on the other hand the concept of migration of services which were a) identified as the fittest for one SME, and b) migrate to another habitat in order to avoid deletion / death. Additionally, the connection to the principle of *Stigmergy* was recognised as it is very close to 1) the model used already in the DBE's version of the EvESim and 2) the concept of Open Knowledge Ecosystems presented at the "'Economic Participation in the Age of Networking -Digital Ecosystems of Knowledge, Business and Services"' Conference at the 07/11/2007 in Brussels[1] is related to the idea of Knowledge-Services in [1]. This relationship didn't influence the work on EvESim, nor the Knowledge-Service ideas but is a direction which could add value to the work-in-progress in the OPAALS project and therefore should be considered in future work.

The following Table 6.1 shows which parts are already implemented at the current stage and which need to be done together with implementations in T3.8, T10.10 and T10.11. The adaptation of the user interface and the comparison of the most suitable infrastructure are not explicitly mentioned in the phase II description of work because they are basic requirements in order to fulfil the goals of the planned activities in T3.8, T10.10 and T10.11.

Already implemented	*To be done*
Replacement of Repast Components	Adaptation of the old user interface (planned within activities in T3.8, T10.10 and 10.11)
Implementation of EvESim components as services	Simulation on a P2P infrastructure vs an application server for performance comparison (planned in T3.8)
Extension of EvESim for basic transactional model simulation	Implementation of the visualisation service
Enhancing simulation performance through code modifications	

Table 6.1: Progress in EvESim implementation The name "Evolutionary

Environment Simulator" orignated from the idea to [1]http://de-2007.eu
provide a simulator for the evolutionary parts of the DBE project. As the simulator turned out to be a framework for visualisation, testing and simulation of interdisciplinary research outcomes, a renaming of the framework is already planned. All the latest information on the simulator can be found at [2].

The following sections are partly taken over from the paper *Peripheral Simulation Framework for Digital Ecosystems* presented at the *1st OPAALS Workshop* on the 26th/27th November 2007 in Rome, Italy. First, an introduction to the Evolutionary Environment Simulator is given. After that, the main limitation of the former approach are summarised. As a consequence of that, a peripheral simulation framework architecture is presented which represents the extended architecture of the new EvESim. Before the concluding remarks on the presented work, the first simulation results are presented as a proof of concept.

6.1- The Evolutionary Environment Simulator

The Evolutionary Environment Simulator, short EvESimulator (see [2] and [3]), was designed and implemented in the DBE[2] project in order to run simulations on a biologically inspired P2P system -the Evolutionary Environment (EvE) (see [4]). Beside the simulation of the EvE the EvESimulator acted as a framework for understanding, visualizing and presenting the DBE concepts to many stakeholders within and outside the DBE project.

As the initial attempt was to do a simple simulation and visualization of the EvE, the simulator was based on the Recursive Porus Agent Simulation Toolkit (Repast) (see [5]). Additionally to a basic framework for agent based simulations, Repast provides the basis for network visualization components. Utilizing Repast and adapting it to the needs of DBE allowed rapid development leveraging an established simulation framework. Nevertheless, drawbacks of using Repast are outlined in Section 6.2.

The objectives of the EvESimulator were adapted and extended during the work with the framework and can be summarized at the end of the DBE project as follows:

Simulation of the intended behavior of the DBE's Evolutionary Environment
Visualization of a network of operational SMEs and consequently the DBE
Facilitation of import and export of data from partners in various formats

Intensification of interdisciplinary collaboration

[2]Digital Business Ecosystem, contract number 507953

Even though the use of the EvESimulator and the integration of stakeholders from business, social-, natural-, and computer science was successful, the modifications of the initial framework lead to problems in performance and extendability which will be summarized in Section 6.2 of this deliverable.

6.1.1. Agent Model

The agent model as well as the service model for the EvESimulator are based on the assumption that a Digital Business Ecosystem (DBE) or a Digital Ecosystem (DE), respectively, is a network of Small and Medium Enterprises (SMEs), each offering and consuming services. We did not take into account explicitly the differences in B2B services versus B2C services but we assumed that the descriptions of the services in this DE will be handled in a business language similar to BML [3] or SBVR [4].

Before describing the model and representation of an actor, respectively, a short definition of the term *service* in the context of the EvE and EvESimulator is given. EvE-Services are meta-descriptions of services which are persisted physically in the P2P network. These meta-description contains service name, basic SBVR description, reference to the actual service and history data of service migration through the network. For detailed information see [8].

Each of these EvE services (in the following just services) are modelled as a vector of attributes in which each attribute has a certain value. These attribute-value pairs identify the service. A very simplified example could be a *hotel room service* which is characterized with the attribute *stars* with a value from 0 to 5, an attribute *price* with a certain value and an attribute *minibar available*, which can be zero or one. The more attributes in the overall system, the more difficult it is to find a perfect attribute match when searching and requesting specific services.

Contrary to the services, the activities of the SMEs are modelled as agents in the simulation. Each of the nodes in the network represent one SME. As regards the simulation we interchangeably use the terms SME, actor, agent and node. Each SME or actor, respectively, has a local service pool (LSP), a list of services on offer (Portfolio), and a list of services on demand (see Figure 6.1).

The local service pool is equivalent to a local repository for services. Besides the SME stating it's clear intentions and interests in the network, the system tracks the SME's service consumption as well. The local service pool data, besides configuration information, is used to set up the environmental conditions of the

[3]Business Modeling Language (BML) was developed in the DBE project for modelling small and medium enterprises based on ontologies. [6]

[4]Semantics of Business Vocabulary and Business Rules (SBVR) is a specification of the OMG. [7]

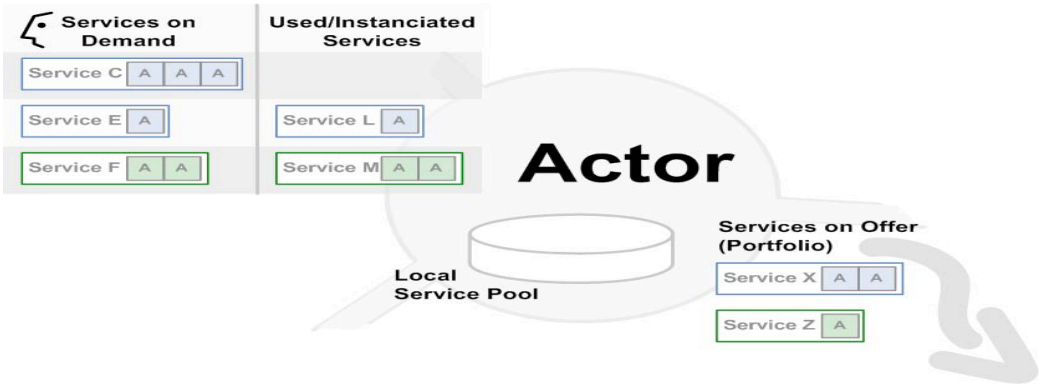


Figure 6.1

services which migrate[5] to the SME. Consequently, services which are probably useful for this SME will reside at the local service pool.

As there can reside just a limited number of services at each habitat or local service pool (LSP), respectively, the LSP has to be thinned out or shrunk as it exceeds a certain threshold. Basis for the currently implemented shrink mechanisms are the list of services on demand. A fitness evaluation names candidates for deletion in order to shrink the LSP. These candidates are then allowed to migrate to other SMEs as long as they don't exceed the permitted number of hops, configurable in the simulator framework.

In the portfolio, the SME manages a list of services which it offers to others. When a new service is on offer, it is simply copied to the local service pool of the producer SME and is distributed to all known neighbors of this SME. Beside the migration to known nodes, a distribution to a random number of new nodes enforces a faster bootstrapping and the mutation capabilities of the network.

The list of services on demand represent also a kind of profile of an SME in the simulation because it indicates the interests of this SME. When new agents are created for the simulation, a list of demanded services is produced as well. Depending on the scheduler of the simulation the SME gets the allowance to request one of these services on demand and looks up the local service pool for adequate services or service compositions. This service search and combination is done by a genetic algorithm and a fitness function, respectively.

[5]Migration in biological terms is the movement of individuals from one biotope or habitat to another. This movement can be driven by better environmental conditions in the new biotope or habitat. [9]

6.1.2. Simulation Examples

In order to see which kind of simulations we want to perform with the new framework introduced later in this deliverable, we sketch here with (1) critical mass assessment and (2) clustering two of the main important simulation cases produced using the EvESimulator.

Critical mass assessment simulations were used to compare performance of global service repositories with distributed clouds of local service pools at SMEs. When an SME in the simulation gets the allowance to request a service, one Genetic Algorithm (GA) is instantiated to search for service combinations in both, the global service repository and the pre-selected local service pool.

The findings on critical mass assessment were the following: After overcoming a bootstrapping phase in which the pre-selection of services in the local service pools is performed, the results of the searches were significantly better in the distributed version. Moreover, it turned out that the total number of attributes in the system for describing the services is an important factor for the performance difference and behavior of the search comparison. Contrary, the differing number of SMEs used in the simulations had no major influence on fitness but on the time of the bootstrapping phase.

In the second simulation case on clustering it was assumed that the network connections will cluster SMEs depending on their service consumption and offer behavior. That should lead to a natural clustering of SMEs and therefore a more effective and self-organizing topology of the network. The most important concepts used were the simile of *migration of services* and the use of a *service history*.

After several simulations it can be concluded that at least for a small number of SMEs the simulated network behaved as expected. Nevertheless, the clustering using the output from the GA based optimization performed much better than the user interaction based approach. For more details on these simulation cases see [8] and [3].

6.1.3. Collaboration Framework

As already mentioned, the EvESimulator works not only as a simulation toolkit but also as a basis for collaboration. Project stakeholders include natural science, social science, computing and also business partners.

One of the core components of the Evolutionary Environment are Genetic Algorithms. It was tried to open the EvESimulator also for future implementations of Genetic Algorithms as well as other optimization methods. A clearly defined application programming interface for optimization algorithms was the basic building block for implementing four different algorithms till the end of the DBE project. Besides that, the model and configuration user interface is designed to allow reconfiguration or even extension of the current framework.

The focus of the efforts for cross-domain collaboration of the EvESimulator was to customize the simulation and visualization capabilities of the framework also for non technically experienced stakeholders. Consequently, a CSV file import for social network topologies was implemented as well as XML-based import and export files. Furthermore, a Java Webstart application provides a one-click installation for the main components of the simulation framework.

As the computing group had to implement the EvE, the EvESimulator acted as the simulation for finding also parameters for the settings of the real EvE. Unfortunately, because of the complexity of matching SBVR models and a lack of resources the EvE could not be fully finished. Additionally, the settings for the EvE strongly depend on the existing structure and size of the SME network and therefore, the parameters are expected to be unique for each application region and also for the evolutionary framework in the OPAALS project. At this point the user-friendly user interface supports an easy simulation of networks and finding of parameters needed to stabilize the network at least in the bootstrapping phase.

From a business perspective the most attractive part of the EvESimulator is probably the visualization component. The simulator comes with a network visualization and a small webserver for showing the key facts about each SME. As the simulator can present different scenarios and the simulation can be slowed down, an SME can easily understand what the concepts of a DE are all about and how the system works based on real-world parameters.

The number of nodes in the networks as well as the number of services in these simulations were quite small. Nevertheless, the time for running these simulations took hours to run on one single PC. The interest of the new simulation architecture has to take these shortcomings into account, which are discussed in Section 1.2 of this deliverable.

6.2- Limitations of the Current Approach

Even though the current EvESimulator model was intended to support the creation of a Evolutionary Environment in a Digital Ecosystem from the beginning on, issues arose during the implementation and simulation. Taking advantage of the interdisciplinary group of stakeholders, the *EvESimulator* now reflects the feature requests of many of them and the model is quite complex.

In this deliverable we evaluate critically the *EvESimulator* for the needs of OPAALS and consequently present a new architecture for overcoming the main issues. In the following the main shortcomings of the *EvESimulator* are summarized:

Too complex model
Limited by using Repast
Lack of performance

Difficulties in estimating the behavior of a real P2P network

In the following we describe these points in more detail:

The initial model described in [3] was modified several times in order to fit to the needs of not only the natural science domain but also to other stakeholders in the project. Although, the *EvESimulator* comes with a transformation method to transform the previous described model into a binary representation for running generic Genetic Algorithms, considerable computing power is needed to do the transformations in large scale networks. In order to be ready for e.g. simulations using real SBVR models it will be necessary to even extend the existing model. It is important to note here, that the existing model is based on the underlying Repast framework model which increases the complexity of the model considerably.

Utilizing the Repast framework for the *EvESimulator* seemed a reasonable approach in the beginning. The main point is that one can model without considering the details of an agent based simulation toolkit and Repast provides a basic agent model, scheduling and visualization capabilities. These reasons are still valid but nevertheless it was figured out that decentralization, customization to the needs of a Digital Ecosystem and encapsulation of our main building blocks (Genetic Algorithm, visualization and configuration) of our model is not easy to achieve. Repast is built in order to support several different types of simulations and as regards the *EvESimulator*, one model would be sufficient for the moment and for the sake of performance it should be lean and open to distribution and simple deployment.

Continuing with the issue of performance, the *EvESimulator* is a stand-alone application which runs currently on one machine. Of course the simulations can take longer or run over night (and also grid computing systems were considered as a solution already), but for a big number of nodes all these approaches seem not very satisfying. As indicated in [3] with the current model and genetic algorithms running in parallel the time span of one simulation with 50 nodes takes about half a day on high-end PC hardware. If we consider large scale-free networks with this setting, it is almost impossible to run the simulations in the current architecture and we have to think of a more distributed approach (see Section 6.3)).

Besides the issues in estimating the behavior of Small and Medium sized Enterprises where we get support from the social science group of the DBE and OPAALS projects (see [3]) it is hard to predict the behavior of a P2P system at work. The availability, network traffic, birth and growth behavior of such a network can be simulated only in a difficult manner under real-world conditions at the moment. Therefore, the settings for the network have to be obtained and related to the P2P network of choice.

6.3- Peripheral Simulation Framework Architecture

In the following we are proposing a *peripheral simulation framework architecture* for a Digital Ecosystem. 'Peripheral' in this case means that many activities in the simulation are distributed in the network like in a peripheral nervous system. The peripheral nervous system in the human body is, opposed to the central nervous system, exposed to external influences like the nodes in a P2P system are exposed to the environment they are executed.

The exposure to external influences and the distribution to a P2P network has some drawbacks for the simulation but has also one important benefit which is: reality. The last issue mentioned in Section 6.2 of this deliverable was the difficulty to estimate the behavior of the actual system and to apply conclusions from simulations directly to the real network. By doing the simulation in a distributed way, the simulation already utilizes the real network and estimations for certain parameters are not longer needed.

Although the peripheral simulation framework could be implemented on any P2P or application server with just a few modifications, it was decided to use the previously described servENT infrastructure. The reason for this is that the servENT P2P network was already developed in the DBE project for the

application in a scale free P2P network for digital ecosystems and so it seems to be the best choice at the moment.

6.3.1. Model and Architecture

As can be seen in Figure 6.2 it was planed to distribute the simulation components over the network and coordinate and appraise the simulation from one node in the network. For the moment this base-node does not need to be implemented as a servENT service but it can be considered to be implemented as a service in the future as well. All the remote components are modelled as servENT core-services and therefore have access to the functionalities of the local services and servent itself.

The base-node of the simulation acts as the central point of coordination and creation. It is planned to take over most of the user interface elements including the import and configuration of the network from the current *EvESimulator*. Consequently, input about the topology and different types of actors within an SME network can be set by a regional social science officer for example.

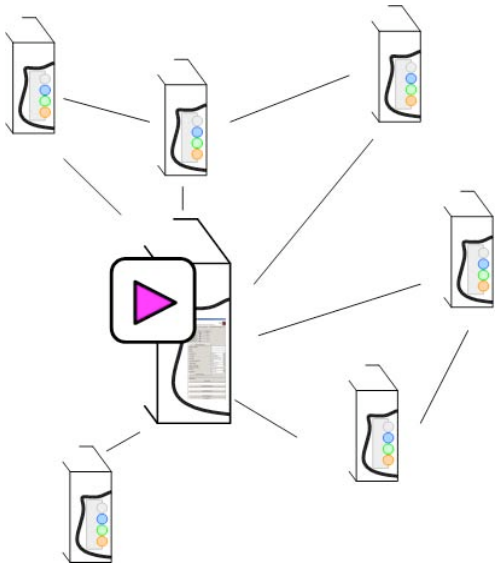


Figure 6.2

Taking the initial settings into account, the node has two main tasks. First, it acts as the creator of the agents themselves. This is achieved by having agent-factories in place as well as service factories. Parts of these factories will be implemented in the agent to enable the agent to create services on it's own. According to user settings, the agents can be created and pushed to the agent pools, distributed over the network (see also Section 6.2). After creating and defining the behavior of the agents, they can act completely independent.

Restricting the total independence, the second task of the servENT is coordination. All the agents are reporting to the base-node on a regular basis. For the sake of coordination, basic adjustment events like tick-speed [6] can be passed to the agents during the simulation as well. Furthermore, the base-node has to observe the reliability of the nodes and check if some of them are offline. Although the servENTs core services have to implement an on-destroy method where final logs can be sent to the basis-node it cannot be taken as granted that these methods will be called in any circumstance.

Important for a coordinator component is also the initial search for servENTs in the environment. Sepending on the uptime of the nodes the base-node can use the servents lookup service to detect the nodes available during the simulation time.

[6]The *EvESimulator* on top of the Repast toolkit is working based on ticks. One tick presents one time step without an assigned time span.

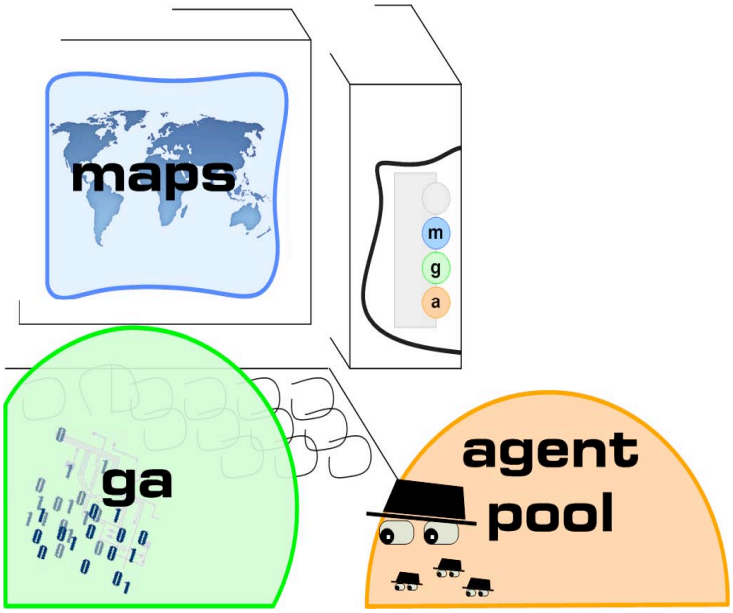


Figure 6.3

6.3.2. Core Services of the Simulation

The idea of the decentralized components of the *EvESimulator* made it necessary to split certain components from the core simulation which need the major part of the processing power. Therefore, three main components are chosen to be modelled as servENT core services (see Figure 6.3). They are namely the agent pool, the Genetic Algorithm (GA) and the visualisation component (Maps).

Firstly, the agent pool acts as registry of agents which act as virtual SMEs. Depending on the configuration, an agent can have a servENT to operate from or several agents can share the resources of an servENT node. This is necessary in order to avoid setting up as many servENTs as nodes in the network which would be too time consuming for larger network simulations. As the agent can access all services on its host servENT, it can also call methods of these services or the servent itself and therefore the servent or certain services can be tested for reliability or performance along a simulation run as well.

The second service is the Genetic Algorithm (GA) service. Based on a common interface there can be several different implementations of GAs which run on the network. Beside the reusability of such a GA service, the distribution of the optimization will bring most probably the highest performance gain of the new architecture. The GA service will be called directly by an agent when the agent gets the allowance to proceed a request. If there is no GA service installed on the same servENT node, the P2P system automatically searches for other GA services in the network and therefore distributes the workload as well.

Finally a visualization of the whole network is planned for the second phase of the OPAALS project. Initially, it was intended to be a visualization UI for the real network we plan to implement the visualization so generic that both, the visualization of the real P2P network as well as the visualization of the simulated evolutionary framework will be feasible. For a better visualization the planned utilization of Google-Maps will give the network an even more realistic appearance.

6.3.3. Additional Changes, Benefits and Drawbacks

Besides the changes in the architecture we learned from the previous experiences in the DBE project that we need to implement a lean simulation framework. That means that the suggested architecture should work without the use of Repast. Even though the implementation of the model, scheduler and agent-structure means additional effort, the benefits of a more flexible framework which can be customized to the needs of a Digital Ecosystem is crucial.

Summarizing the benefits of the new architecture and having the limitations of the old version in mind, the new *Peripheral EvESimulator* will succeed through increased performance and due to the more realistic network behavior. The decision of encapsulating the three main components and the reuse of the upcoming visualization considerably enhances the re-usability and the future implementation of an evolutionary framework. Through the utilization of the actual network we learn more about possible customers needs and issues when programming services and the network is already tested during the simulation.

Nevertheless, there are still issues which have to be considered in future implementations. The model is still complex and if at a later stage the natural language based descriptions, e.g. via SBVR, will be tested as well, the model becomes even more complex. In order to overcome parts of that complexity, agents and services need to be modeled in a generic way so that new service descriptions can be used without taking into account or inheriting the legacy attributes of previous models. Furthermore, we made heavy use of the Java interfaces to make extensions at a later stage easier. Another big issue is the 'peripheral' one. The simulation can be effected by all the external influences as the real network. Consequently, nodes can go offline and the loss of this data or at least a recognition of this loss has to be considered when designing the log and communication behavior.

6.4- First Simulation Results

The following three simulation cases are a proof-of-concept for the new and distributed simulation framework. As the main focus of this work is concerned with providing a cross-domain simulation framework, the simulations themselves are just to show applicable simulation cases. Beside the two simulation cases of the DBE, namely Critical Mass and Clustering, we introduce one more for the simulation of a basic transaction environment. This simulation case is not intended to provide deep-grounded answers to research questions for the design of a transactional model but should show the first step in introducing transaction simulations in the simulation framework.

Before the simulation configurations and results are described, a view remarks are done on the issues and drawbacks we faced during the implementation of the intended software and its behavior.

6.4.1. Issues and Drawbacks

One of the most important reasons for distributing EvESim over a P2P network were the performance issues with the old and centralised system. The change of the code base and the split into basic services was a considerable effort because a complete redesign was necessary. Components which were used from the previously used Repast Simulation Framework had to be implemented from scratch and others had to be adopted to the changing requirements.

As mentioned before, we planned to reuse parts of the servENT P2P system from the DBE project in order to distribute the simulation. During the deployment on a network of servENTs in a LAN, some issues arose as regards the communication of nodes and opening and closing connections at a high frequency (1 service call per SME agent each 10 seconds). With a network of 100 SMEs that lead to 10 service calls per second and resulted in various heap overflows. Although, TechIdeas provided a new release of the servENT, the exact reason for the problems couldn't not been identified until this deliverable was due, so

we decided to run the services locally on one machine for the moment. The EvESim framework will be further extended in Phase II of the OPAALS project and there are also some performance tests planned for further performance improvements.

In order to run larger scale simulations on the service based EvESim, locally, we simplified the Genetic Algorithms and performed targeted optimisations of the code structure. Details can be seen in the JavaDocs in the EvESim codebase [2].

Furthermore, the first simulation results indicated that the results in larger scale networks depend heavily on the distribution or seeding of the services in the local service pools. Here is a high optimisation potential for the overall performance of the system. Possibly, the initial distribution and bootstrapping should be adapted to the scale of the network in order to get better results. Nevertheless, there is still the need for detailed simulation and observation of the influence of parameters like, e.g. 1) number of services in a local service pool, 2) number of useless hops until deletion, 3) migration triggers or 4) social capital[7], to the network behavior and scaling.

6.4.2. Critical Mass and Clustering

In the following the first results of the redesigned Critical Mass assessment can be found. In addition to the simulation cases described in [8] and [3], the simulations of critical mass and clustering can be performed now simultaneously. That means that the critical mass assessment is configured as a network of, e.g. two groups of SMEs, where one group is offering services, which are demanded by the other group. In the former simulations, the services have been distributed randomly over the network. Now we have more influence on the offer and demand behavior of the SMEs and therefore, a more targeted simulation can be performed.

Additionally, the reporting facilities were enhanced so that during one simulation, both, the critical mass analysis and the clustering can be observed. We performed 6 simulations for getting the first simulation results of critical mass and clustering.

Parameter	*Value*	*Short description*
NoOfSMEs	5	Number of SMEs per group
SMEGroups	2	Number of groups in the system
NoServicesOff	5-10	Number of Services on Offer
NoServicesOffTot	al50	Total number of Services on Offer in the simulation network
NoServicesDem	5-10	Number of Services on Demand
MultOffDem	3	Demanded-service-length to offered-service-length ratio
AttPerService	5	Number of attribute/value pairs for identification of the service
NumAttributes	20	Total number of different attribute/value pairs in the overall system
SPTresholdFact	1	Factor for increasing the Local Service Pool size (== max 15 services in the LSP)
GA call	5-15 [s]	GA call each x seconds
SimTime	2000 [s]	Simulation Time in seconds

Table 6.2: Main configuration parameters for simulation case (1).

Simulation case (1): Table 6.2 shows the basic simulation parameters for the first simulation case. Starting with these parameters we varied some settings and [7]Modelled in the simulation as the number of connections in a social network analysed the results. The initial service distribution to the local service pools is random in all simulation cases presented here. Simulation case (1) was done with a very small network of 2 groups with 5 SMEs, each. As can be seen in Table 6.2, the simulation was running for about half an hour. The results showed that the optimisations of the central service pool (CSP) resulted in slightly better results than the optimisations on the local service pools from the beginning on. As concerns the clustering, the producer-consumer edges were identified correctly, whereas the benchmark edges lack in a clear clustering in two groups. The reason for that could be whether the small number of services in the system or the slow propagation of history information coming from former usages of a service.

The search result of all simulations was selected based on a basic fitness threshhold of 70 percent of fitting attributes of request in comparison to the selected solution. If the fittest service combination selected by the genetic algorithms didn't have a match of at least 70 percent it was not instantiated and the SME requested the same service later again. In order to check if this threshhold is essential for the result of the simulation we ran several simulations without this threshold which resulted in no considerable change in comparison to the simulations with the threshold.

Simulation case (1.1): In this simulation case we modified the number of services in the local service pool. By increasing the SPTresholdFact factor up to 10, each LSP held a maximum of 150 services in comparison to the central service pool with 50 services. The result showed that initially the optimisations on the local service pools performed equal or better than the one on the CSP. The longer the simulation ran, the more services migrated to the local service pools and therefore the local optimisation became slightly worse than the central one. In the clustering behavior we recognised no differences to the simulation case (1).

Simulation case (1.2): Here we modified the number of groups in comparison to simulation case (1). Instead of 2 groups with 5 services each, we simulated a network with 5 groups and 2 SMEs each. Beside the fact that the fitness of the optimisation results was in general far below the one from case (1), the optimisations on the central service pool was much better than the ones on the local. As the network was heterogeneous in terms of service offers and types of SMEs, the clustering of the benchmark SMEs was worse as well. Obviously, that was also the result of the optimisation results with low fitness. As the services are migrating just if an SME decided to use a service and this service has to reach at least a fitness of 70 percent, service combinations with low fitness slow down the distribution of the services in the network. Consequently, more work on bootstrap mechanisms for such networks is needed.

Simulation case (1.3): In order to show the difference having more heterogeneous services in the system we multiplied the number of services by the factor of 10. That leads to more diverse services in the system. Although, the migration of services in the network lead to an increasing performance of the local optimisations, the performance of both, local and global, was much lower than in simulation case (1). The numbers of clustering showed no indication if the clustering worked in this simulation case. As the network was too diverse it was not possible to cluster the network within the given time.

Simulation case (2): This simulation case was done with 2 groups and 50 SMEs in each group. In order to adapt the local service pool size to the number of different services in the overall system, the SPTresholdFact was increased so that each SME had a local service pool of about 10 percent of the size of central service pool. With these modification in comparison with simulation case (1), the local and central optimisations showed almost exactly the same performance. Although the results from the genetic algorithms disperse considerably from one result to the other, the trend lines are at a constant hight and for local and global optimisations and especially the producer-consumer relationships were identified correctly, with more than 90 percent probability. Nevertheless, the recognition of similar SMEs by analysing the service histories was very slow and reached at the end of the simulation just slightly more than 50 percent.

Simulation case (3): This last simulation of critical mass and clustering we simulated with 2 groups and 100 SMEs each. The local service pools were again about 10 percent of the central service pool size. The results were considerable better than all the others before. After a short period, where the services migrated

to the adequate LSPs depending on the GA results, the local optimisations performed much better than the centralised. Figure 1.4 shows the results of this simulation including the trend lines for local and centralised optimisation. Beside the good results in critical mass, also the clustering results were much better than in all other simulations. With extending the capabilities of the simulator in a fully distributed simulation framework even larger scale simulations will indicate the behavior the scale free properties of the system can be further elaborated.

6.4.3. Transactions

The theoretical work on transactional networks, documented in deliverables [10] and [11] will be simulated and tested in T3.8 in phase II of the OPAALS project using and extending the EvESim framework. In the following, an implementation of the basic components for a Virtual Private Transaction Network (VPTN) simulation can be found. We tried to set up a very generic and simple simulation with serial and parallel[8] transactions and get indicators for a service execution behavior without exacerbated behavior.

For the implementation of transactions we implemented a adapted version of

[8]Two or more services executed in parallel.

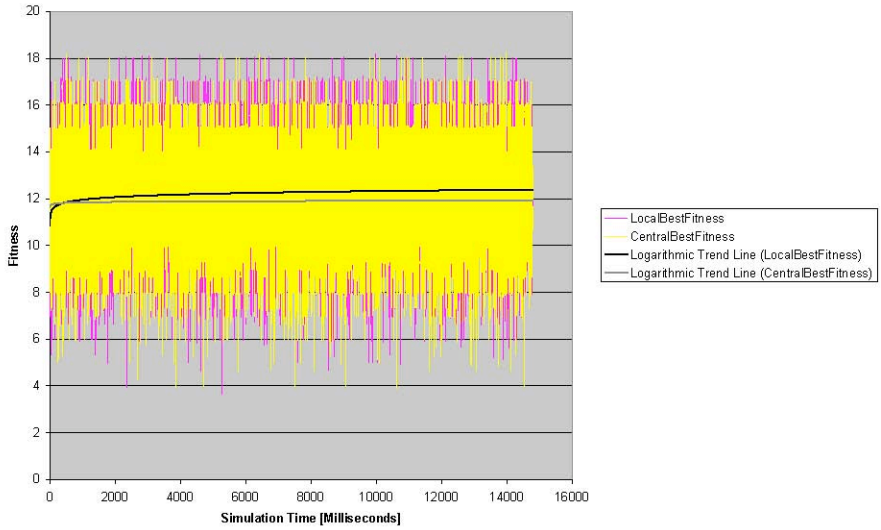


Figure 1.4: Results of critical mass assessment with 2 groups and 100 SMEs.

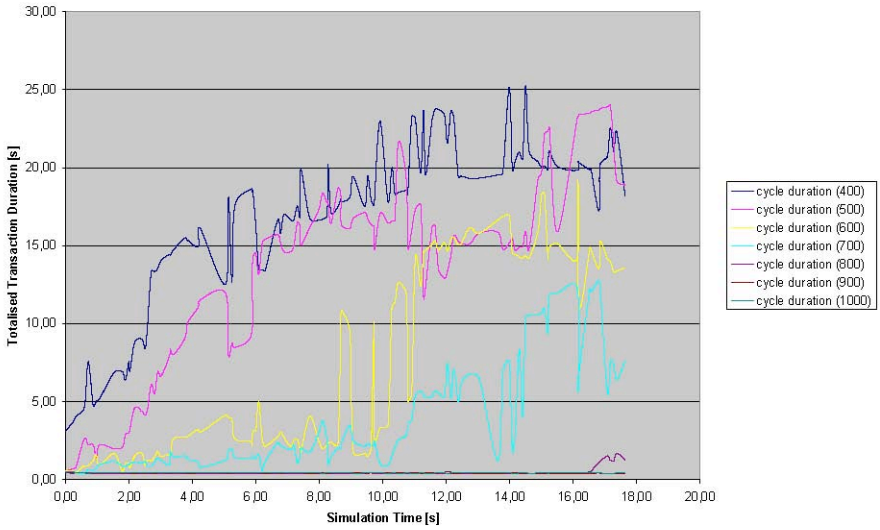


Figure 1.5: Result of the first transaction scenarios (execution time per service = 100 Milliseconds).

the agents. Each of the 10 agents in the simulation got one transaction tree for execution. The service executions within the execution tree are simulated with a variable sleep() time. Consequently, if a transaction is called by an agent, the services are called in serial or parallel order. The duration for the execution of the full tree is documented and represents the basis of the subsequent analysis. If two transaction trees are requesting the same service, the requests have to wait until they get a notification when the service is free again and one of the pending service calls can access the respective service again.

We wanted to find out which is the relation between the average execution length of the services in the system and the call frequency for the transaction trees in order to execute all transactions without building up the call queues at each of the services. In the following the results of the first simulations are discussed.

The simulation ran with one transaction tree in order to keep it simple at the beginning. The transaction tree includes just serial and parallel service calls of 5 services. Each of the 5 SMEs executes the same transaction tree with varying starting points. The sleep time, representing the service execution was 100 Milliseconds for each of the services in the tree. As can be seen in Figure 1.5, the execution of the services was stable up to a cycle duration of about 1000 Milliseconds. That means that the system in this configuration is stable as long as the transaction trees are called less often than each 1000 Milliseconds.

At the moment the queuing of the service calls are programmed as synchronized java method calls. The results therefore are not equivalent to the behavior a application server or P2P system would deal with

simultaneous service calls. Therefore, the results shown in Figure 1.5 just show that the modifications in the simulation framework are running as expected. Consequently, in task T3.8 the work will be extended in order to call real services on P2P nodes or/and application servers in order to make the simulations more realistic. Additional, the transaction trees will become dynamically loadable from XML files.

6.5- Concluding Remarks and Future Perspectives

Based on the EvESim framework of the DBE which was recognised as a very challenging and promising approach for interdisciplinary research, this chapter shows how the limitations of DBEs EvESim can be equalised. Moreover, the new distributed EvESim framework can satisfy the needs of OPAALS and offers a couple of new abilities for future work.

With the integration of many disciplines in the evolution of the EvESim framework we go a new way of building innovative simulation frameworks. Although, simulations on particular problems can be done on specialised simulation software within each discipline, the outcomes and findings are collected in the EvESimulator and other partners benefit from them. Here a short example: The simulation of social networks in a region lead to a couple of parameters which are essential for the behavior of the actors within this network. These parameters will be included in EvESim. In parallel the computing group implements a P2P network on which the DBE will run in the future but as there are no users available yet, they cannot test if its really performing according to the needs of the social network where it will be used. With the EvESim framework findings on behavior of actors can be combined with service calls and processing in the simulation and therefore, the two groups can benefit from each other. Nevertheless, the social science team will run simulations and the computing group will implement test cases before the outcomes can be integrated in the EvESim framework.

Although, the EvESim in its new, distributed and component based version can be tested and extended now according to the needs of the project, it needs still a view modifications which will be implemented mainly in task T3.8 and T10.11. Additionally, the work on extending the test of the transactional model as well as the integration of the P2P infrastructure visualisation from task T10.10 are on their way. Especially with the integration of the P2P infrastructure visualisation it is possible to visualize both, the actual P2P network and the simulation network with the same map-based technologies.

6.6 References (for Chapter 6)

[1] Raimund Eder, Thomas Kurz, Davide Joss, Antonella Filieri, Miriam Pezzuto, Hagen Peukert, Alexander Marios, Stan Hendryx, and Thomas Heistracher. D2.2 -Automatic code structure and workflow generation from natural language models. Open Philosophies for Associative Autopoietic digitaL ecosystemS (OPAALS) Project.

[2] Thomas Kurz, Markus Duerr, Christian Ehgartner, and Roman Greil. Evolutionary Environment Simulator -evesim. Webpage. http://evesim. sourceforge.net. Last accessed on 12/22/2006.

[3] Thomas Kurz, Giulio Marcon, and Antonella Passani Hisanaga Okada, Thomas Heistracher. D9.2 - Report on Evolutionary and Distributed Fitness Environment. Digital Business Ecosystems Project, July 2006.

[4] Claudius Masuch. Evolutionary Environment Network. Webpage. http: //evenet.sourceforge.net. Last accessed on 12/22/2006.

[5] Nick Collier, Tom Howe, Robert Najlis, Michael North, and Jerry R. Vos. Recursive porus agent simulation toolkit -repast. Webpage. http:// repast.sourceforge.net. Last accessed on 06/22/2006.

[6] Maurizio de Tommasi. D15.1 -Business Modelling Language 1.0. Version 1.0, April 2005.

[7] OMG. *Semantics of Business Vocabulary and Business Rules Specification*, March 2006. First interim specification, http://www.omg.org/docs/ dtc/06-03-02.pdf. Last accessed on 12/22/2006.

[8] Thomas Kurz and Thomas Heistracher. Simulation of a Self-Optimising Digital Ecosystem. *IEEE First International Conference on Digital Ecosystems and Technologies*, 2007.

[9] Mike Begon, John Harper, and Colin Townsend. *Ecology: Individuals, Populations and Communities. Third Edition*. Blackwell Publishing, 1996.
21

[10] Paul Krause. D3.1 -Preliminary architecture for autopoietic P2P network focussing on hierarchical virtual super-peers, birth and growth models. Open Philosophies for Associative Autopoietic digitaL ecosystemS (OPAALS) Project, July 2006.

[11] Paul Krause. D3.2 -Report on formal analysis of autopoietic P2P networks, together with predictions of performance. Open Philosophies for Associative Autopoietic digitaL ecosystemS (OPAALS) Project.

7. Distributed Transactions over Mobile Networks (CN)

Due to the increasing computing, processing and communication power available on mobile devices, we are witnessing a gradual shift of the modus operandi from the traditional workstation-based computing environments to fully mobile ones [12],[13]. Thanks to their promise of increasing users' flexibility and speeding up interactions, users are adopting more and more their personal devices for carrying out businesses, or, more in general, for retrieving and storing any information that is needed during their daily activities. As an example, this information can include their shared calendar, distributed databases, etc. This technological trend is opening up vast opportunities for developing mobile collaborative applications where, without the need for any dedicated infrastructure, it becomes possible to carry out businesses while being on-the-move.

However, before moving from a fixed network infrastructure to a fully distributed computing environment, there are several technological challenges that need to be addressed [15]: present connections can only rely on a limited bandwidth, intermittent connectivity represents a rule rather than an exception, nodes mobility highly impacts the performance of any protocol, energy constraints the operations that can be performed, scalability affects the number of devices being part of the network [14]. Clearly, well understood problems need to be completely revised in light of these new scenarios, and existing solutions need to be adapted to these new constraints.

This is also the case of distributed transaction processing systems [19], in which applications utilize shared computing resources, e.g., restaurant reservation, a transfer of funds from one account to another, telephony, etc. A mobile transaction is defined as a "*set of relatively independent (component) transactions which can interleave in any way with other mobile transactions*" [17]. In general, a mobile host is characterized by limited energy resources, physical mobility, intermittent connectivity and limited bandwidth. Clearly, this poses completely new challenges to the support of mobile transactions, especially when coming to the ACID properties that transactions are supposed to verify: Atomicity, Consistency, Isolation and Durability. Traditional transaction models, built with a database-centric view, are not able to accommodate the new constraints posed by mobile computing environments [23].

In the following, we will review a set of mobile computing scenarios, with the technological challenges related to the support of distributed transactions. In particular, as discussed in D3.1, the network topology has an extremely relevant role in determining some of the key system properties, such as redundancy, robustness, etc. Such properties are of paramount importance in order to ensure that the network verifies the key requirements of digital business ecosystems, e.g., no critical dependencies, no critical point of failures and being fully distributed. We will then analyze what are the key topological properties of the mobile networks considered, and propose possible approaches in order to let some specific characteristics emerge.

7.1- Mobile Computing Environments

A mobile computing environment is generally composed by Mobile Nodes (MN) communicating with each other and possibly with a fixed infrastructure. Depending on the specific application scenario considered, this can happen in different ways. It can be that MNs are single-hop away from the fixed infrastructure, but also that no fixed infrastructure is available at all. In the following, we will analyze the technological challenges that need to be addressed in order to support the execution of mobile transactions. In particular, we will present three different mobile network architectures in order of increasing complexity: infrastructure based mobile networks, mobile P2P networks and Delay Tolerant Networks.

7.1.1. Infrastructure-Based Mobile Network

The simplest case is that of MNs being a single-hop away from the fixed infrastructure. In this case, mobile devices are connected to the high-speed wired network by means of Base Stations (BSs), which manage the communication to/from MNs. This is the case of cellular networks, where mobile phones are accessing, through their wireless network interface, resources residing on the fixed infrastructure (Figure 1).

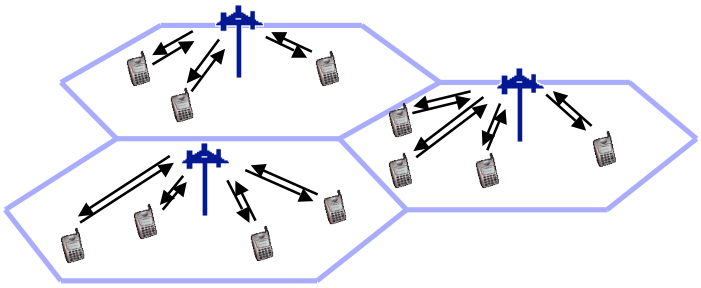


Figure 1: Infrastructure based mobile network.

Clearly, connectivity is available as long as the mobile host resides in the area served by the base station. When a mobile node moves outside the area served by a base station it can occur that (i) the node enters a new cell (ii) the node moves into an unserved area. The former case leads to a handoff [18], where the node will be attached to the new base station, while the latter leads to a temporarily disconnection, where connectivity is temporarily unavailable. In the case of handoff, any communication occurring with the mobile nodes is re-routed to the new base station. Clearly, this occurs without any disruption to calls or data transfers performed by the mobile node during the handoff.

The application scenarios included in this network configuration comprise mobile commerce transactions, remote working, and distributed planning.

7.1.2. Mobile Peer-to-Peer Systems

A second case is that of Mobile Peer-to-Peer Systems [9], [16], where MNs compose a network on-the-fly, without the need for any dedicated infrastructure. In this case, any communicating device (i.e., PDA, smart phone, laptop, etc.) can potentially become part of the communicating superstructure, and join/leave the network at any time. Appropriate self-organizing mechanisms need to be in place in order to let newcomers join the network, and to establish and maintain communications among hosts. Nodes can potentially be mobile, thus leading to temporarily partitions of the network whenever a subset of the nodes moves outside the mutual communication range. In addition, these systems are by definition fully distributed, since it is not possible to identify an always-on central server, thus making any central control difficult, if not impossible.

The application domains covered by these networks include disaster application scenarios (e.g., fire protection, storm and flood management), intelligent transportation systems (e.g., traffic management, etc.), enterprise applications (i.e., fleet management).

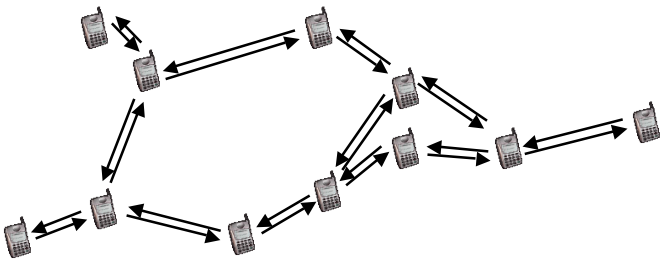


Figure 2: Mobile Peer-to-Peer System.

7.1.3. Delay Tolerant Networks

A last case is that of Delay Tolerant Networks (DTNs) [1],[2] where connectivity is not taken for granted and mobility is exploited (by means of opportunistic forwarding schemes) to provide network-wide communications. In order to diffuse information in the presence of a disconnected topology, a *store-carry-and-forward* mechanism is typically employed at each node. As depicted in Figure 3, this consists in nodes temporarily storing messages not destined to themselves, and forwarding these messages to other nodes encountered while moving. This process is iterated until the final destination is reached. As an example, in

Figure 3 the bus is in charge of collecting data that was temporarily stored at intermediate nodes. Such data is then gathered and subsequently delivered either to another intermediate node encountered along the path, or over the Internet, when reaching the city.

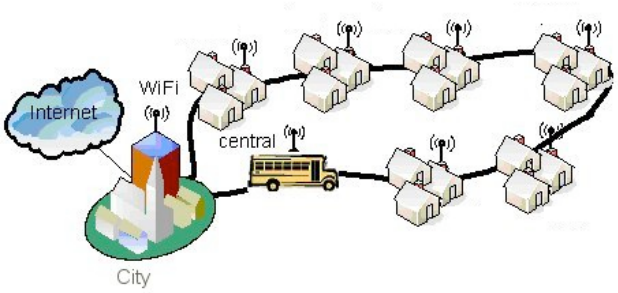


Figure 3: Delay Tolerant Network.

The diffuse interest in these network architectures arises from considerations relative to scenarios where the devices may be disconnected due to some physical constraints. Thanks to the constantly increasing power of personal mobile devices, this scenario is becoming widely diffused in today's society. An example of such scenarios when buses and other public transportation means are used for carrying and disseminating information [8], [9]. Such systems may be used, e.g., to diffuse information about traffic situation, parking availability, special events (conferences, fairs etc.), local advertisement, video-surveillance and similar tasks. In these situations, connectivity occurs rarely and with no predefined schedule. Clearly, this determines the need to define an architecture able to handle disconnected operations. Another example includes the case where the mobility of vehicles is exploited to serve rural areas that can not be reached by other means [11]. Another example is that of social computing, where mobile devices are exploited in order to interact with neighboring peers, and occasionally exchange data such as business cards or music. Finally, the most futuristic case is interplanetary Internet [7], where the planets' orbits drive the presence/absence of line-of-sight and hence the possibility of communications.

7.1.4. Distributed Transactions in Mobile Computing Environments

The intrinsic dynamic nature of mobile networks makes the support of distributed transactions an extremely challenging task. This is mostly due to the limited adaptability of existing transaction models to deal with variable network conditions such as intermittent connectivity, variable bandwidth, nodes' mobility and heterogeneity.

In mobile environments, most of the devices have limited available resources such as processing power, storage, energy, etc. Further, intermittent connectivity will occur due to the intrinsic node mobility, and the network will often become partitioned, especially in the case of mobile ad-hoc networks (MANETs) and DTNs. This will influence the ability of nodes to (i) provide services (ii) delegate operations to other nodes. As a consequence, the availability of resources is constantly changing over time, since nodes are intermittently leaving and joining the network. This uncertainty of resources needs to be handled gracefully, in order to support adaptive ACID properties.

A possible approach to deal with such dynamic behavior is to add redundancy to the network, such as replicating resources and services. Replication of information aims at disseminating copies of a certain resource over different peers of the network. Clearly, one of the major issues in such an approach is how to efficiently maintain the consistency of the disseminated copies. Data dissemination protocols are then implemented in order to efficiently trade off the availability of information with the resources that are needed for running such algorithms. Another example of redundancy is that of shared logs, that are used for recovering from failures. As an example, in [21], a Shared Log Space has been proposed in order to improve data availability when running distributed transactions. In the case of a node failure, the neighboring nodes keep track of the status of the transaction, thus facilitating a subsequent recovery. Clearly, the application of these techniques highly depends on the level of dynamicity of the network. For relatively highly mobile networks, the cost incurred in maintaining consistent the different replicas of resources and services becomes too prohibitive for the limited resources of mobile devices.

Intermittent availability of resources, especially connectivity, poses great challenges to existing transaction models. As an example, we can think of a node initiating a transaction, and disconnecting from the network before terminating it, and reconnecting later on. In this case, it is possible to (i) abort the entire transaction (ii) define efficient techniques for recovering such a disconnection, without the need to restart the transaction from scratch. This latter case can be achieved by means of (i) long-lived transactions (ii) nested transactions. *Long-lived transactions* refer to transactions lasting much longer, if compared to transactions occurring in fixed infrastructures. However, large long-lived transactions are extremely inefficient when running as a single transaction, since they cannot take advantage of partially-completed work. This problem is alleviated by means of *nested transaction models*, in which a top-down approach is used to decompose the original complex transaction into smaller ones. In this approach, individual sub-transactions can be committed separately, and partial results made available to other running transactions. However, how to efficiently partition a larger transaction into smaller ones is an open question. As an example, a transaction should be decomposed into smaller sub-transactions, with the more resource-intensive operations possibly assigned to hosts residing on the fixed network infrastructure, for which no constraint exists in terms of bandwidth, computing power and energy. Further, when a node is in charge of a sub-transaction and loses connectivity temporarily, it continues the processing while being offline, and updates the coordinating node on the outcome of its processing as soon as it becomes again online. Also in this case, efficient replication techniques are needed in order to maintain the consistency of the processed data between online and offline processing. Clearly, also in this case there is a trade-off: the more the replicas, the larger the overhead needed for maintaining consistent information. This trade-off can be specified in the transaction behavior. As an example, in the Mobile Trans middleware [43], nodes specify their behavior in a transaction policy XML declarative file [44]. Such file defines the specific transaction properties such as transaction tolerance time and consistency level, which represent the two parameters according to which it is possible to trade consistency for resources.

Specific coordination algorithms need to be devised in order to support the coordination of the different peers involved in a distributed transaction. In particular, the traditional 2 Phase Commit Protocol needs to be extended in order to support the constraints imposed by mobile networks. In fact, mobile nodes are expected to process data offline when losing the connection. This needs to be taken into account when the nodes connect back to the network. Working in isolation for prolonged period of times often lead to a large number of possible conflicts, with a consequently high abort rate. This calls for innovative techniques in order to mitigate this problem and to maintain a sufficiently high commit rate despite the intermittent availability of resources. It is therefore important to define *disconnection-tolerant* and *partition-tolerant* commit protocols, in order to achieve the right balance between long blocking, and cascading abort. As an example, this is addressed in HiCoMo [20] by means of aggregation models. This is to be coupled with nested transaction models that are needed in order to permit parts of a transaction to fail, without necessarily aborting the entire transaction.

Finally, security is a critical issue in mobile environments, since the network is established on the fly, without any prior knowledge of the participating nodes. In principle, it is not possible to assume third parties guaranteeing the credentials of mobile nodes, or the possibility to have certificates securing communications and transactions among nodes. Hence, security threats represent a serious problem, especially when running business applications. This should be explicitly taken into account when designing distributed transaction systems over mobile networks.

All the aforementioned issues assume a different relevance depending on the mobile network considered. In infrastructure-based mobile networks disconnections may occur, but not as frequently as in MANETs or DTNs. As a consequence, the need to support long-lived transactions is not critical, although it has to be supported. In cellular networks it can happen that a mobile node moves through different base stations while running a transaction. It is therefore important to incorporate movement behavior as an intrinsic feature of such systems, and seamlessly support the capability to move the transaction state from one service cell to another. The Kangaroo Transactions [21], has been proposed as a solution to this problem in the case of a multi-database environment.

In Delay Tolerant Networks the disconnected modus operandi represents the rule rather than an exception. Similarly, transactions are expected to last very long in DTNs, where communication occasions may be very rare depending on the specific application scenario. However, differently from a MANET network setting, for most DTN application domains (i.e., bus networks, satellite communications) the connectivity is not random, but can be predicted a priori. This can be exploited in order to define efficient temporal nested transaction models, in which sub-transaction are assigned to nodes depending from the specific instant in time.

Due to the store-carry-and-forward paradigm applied in DTNs, severe security threats need to be taken into account. In particular, nodes are delegating other nodes the delivery of messages. This means that trust mechanisms need to be in place in order to guarantee the trustworthiness of the nodes to which we are forwarding messages. In addition, appropriate mechanisms are required to ensure a reliable completion of a distributed transaction. This could relay on the custody transfer mechanism available on DTNs. This mechanism consists in reliably moving the responsibility of the delivery of a message from one node to the other [25].

7.2- Autopoietic P2P Mobile Networks

7.2.1. Infrastructure-Based Mobile Network

Infrastructure-based Mobile Networks consist in a group of mobile nodes served by a central coordinating entity called the *base station* (BS). The base station arbitrates the channel allocation between mobile nodes [31], and takes care of any maintenance operation needed to preserve nodes connectivity.

7.2.2. Supporting Autopoietic P2P Networks in Infrastructure-Based Mobile Networks

In order to support an autopoietic P2P network topology in infrastructure-based networks we need to form an overlay wireless network that exhibits scale-free characteristics and self-organizes into a load-balanced state. Decomposition of this self-organization capability in the form of guided topology generation explores the applicability of the small world and scale-free concepts to the case of infrastructure based mobile networks. The important issue is to investigate how such concepts (scale-free and small world) can be implemented in cellular wireless networks.

Here the goal is to adapt existing network infrastructure in order to achieve a network topology (or an overlay network topology) similar to the one required for supporting distributed transactions. However, a pure cellular network presents a limited level of flexibility, being constrained by the underneath base stations placements. Indeed, BS station placement typically follows the need for connectivity rather than an accurate topological planning. For this reason, a static fixed network infrastructure can hardly be adapted to the requirements of autopoietic P2P networks.

In order to overcome such limitation, an emerging trend is to combine the cellular network model with the mobile ad-hoc network (MANET) one. As explained in more detail in the next section, in a MANET network sources and destinations communicate with each other through the use of multi-hop paths along the different peers of the network, and without the aid of a backbone infrastructure. Such ad-hoc mode of operation allows all wireless devices within range of each other to discover and communicate in a peer-to-peer fashion without involving central access points. With this concepts, data-rate is increased [32] and transmission power is reduced [33]. Other properties of this combined infrastructure are enhanced network capacity [34], better load balancing [35] and also extended coverage [36].

If we analyze the conceptual model of these two network architectures, it is observable that cellular network model is centralized whereas a mobile P2P based network model is fully distributed. A possible approach for combining these two modus operandi is to adopt a joint cellular and fixed relay network such

as the integrated cellular and ad-hoc relay (iCAR) system proposed in [40, 41, 42]. In iCAR, it is assumed that, in addition to the traditional wireless infrastructure (i.e., cellular network), a network of relay nodes is deployed in order to achieve better system performance and a dynamic load balancing of the data traffic. Such integrated network will be composed, in addition to BSs and mobile nodes, by Fixed Relay Nodes (FRNs) that are wireless communicating devices that can communicate directly with a mobile node, a BS, or another FRN via different air interfaces. FRN operate in either the cellular band or unlicensed bands. FRNs do not have any wired connection, and their role is to forward user data either to the next FRN, or to a base station. The proposed system architecture is depicted in Figure 4. In [42] it is shown that by assuming a cellular network with a fixed topology, and by assuming a certain freedom in the placement of RNs, it becomes possible to reproduce the *preferential attachment rule* and obtain a network with a small average path length and high clustering coefficient (Small World).

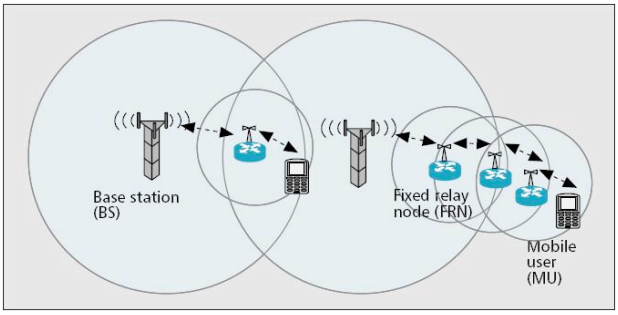


Figure 4 Example of a Wireless Access Network.

The model described before allows integration of an existing cellular network with relay nodes, in a way such that the resulting network topology presents a high clustering coefficient and a power law degree distribution. The drawback of such an approach is that it requires additional infrastructure (RNs) and the possibility to freely place such additional nodes. However, in many practical situations this is not possible. An alternative approach consists in letting the mobile nodes themselves acting both as relay nodes and as nodes connected to the cellular network. Such an approach is applied to the design of the *Sphinx* system [31], with aim of achieving an optimal use of both cellular network and the mobile ad-hoc network. The performance evaluation of the proposed system shows that it is possible to improve throughput and energy consumption, together with the fairness in the channel utilization and resilience to the nodes mobility.

7.2.3. Mobile ADHOC Networks

Wireless multi-hop ad hoc networks consist of a collection of (possibly mobile) nodes, which are distributed over a spatial region and dynamically form a temporary network in a fully decentralized fashion.

When considering P2P overlays in MANETs, we are constrained by the underneath network topology deriving from the physical connections among nodes. The major parameter that can be tuned in order to dynamically vary the network topology is the transmission power, and consequently the communication range, and the nodes density.

Following a simple propagation-receiver model, a node *i* is able to listen to a transmitting node *j*, if the transmission power relative to the noise exceed the signal-to-noise ratio SNR:

$$\frac{P_j / r_{i,j}^{\alpha}}{noise} \geq SNR$$

where P_j is the transmission power of node *j*, $r_{i,j}$ represents the distance between node *i* and node *j*, and α represents the path loss exponent. This formula relates the transmission power to the background noise and the distance between two distinct nodes.

Clearly, communication range is strongly related to nodes density. Since in an ad-hoc network nodes will communicate by means of multi-hop communications, a larger nodes' density will result in stronger network connectivity.

7.3- Supporting Autopoietic P2P Networks in MANETs

As discussed in the previous section, the transmission range represents the primary parameter used for *tuning* the network topology. In Figure 5, network topologies are presented in the case of 75 nodes deployed over a 600 x 600 m square playground and a communication range of 50, 75 100 and 125 m, respectively. As it is intuitively clear, as the communication range of nodes increase, a higher number of connected components appear in the playground.

In the example described above, all nodes obey to the same rules, adopting a uniform transmission power and therefore an equal communication range. In this case we can easily achieve a strong connectivity for a sufficiently high communication range [29]. Typically is the communication range exceeds a certain threshold, a unique giant connected component appears in the network.

This first model, while being extremely simple, represents some clear drawbacks. First, it is based on a global rule and, as such, nodes need to be all aligned on the transmission power to be applied. Second, the scheme can not be adapted to non-uniform nodes placement. In the case of irregular deployments, there could be more dense regions, for which a lower transmission power (and therefore smaller communication range) suffices for having a sufficiently connected component, and more sparse one, for which a higher transmission power is needed in order to reach remote nodes.

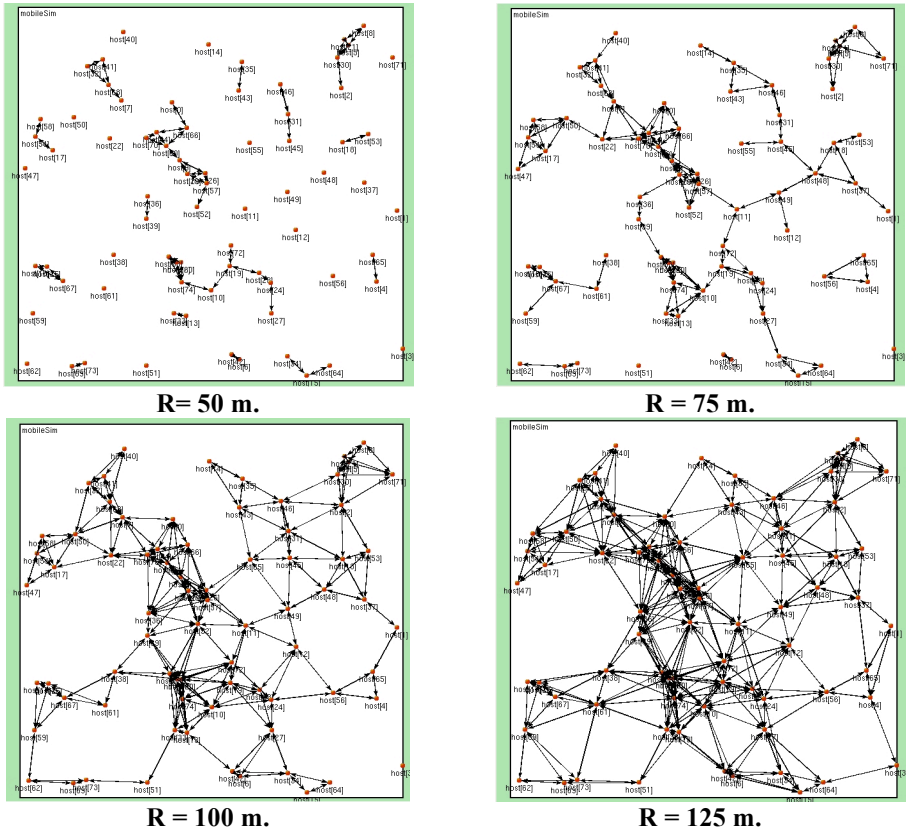


Figure 5: Network topology deriving from 75 nodes deployed over a 600 x 600 m playground, with a communication range of 50, 75, 100, 125 m., respectively.

In order to overcome the limitations of the fixed transmission power model, it is possible to assume that nodes are exchanging information messages in order to explore the connectivity pattern of neighboring nodes. This is achieved by means of simple "hello message" exchanges. This additional degree of freedom lets us achieve a more flexible network topology. In this case, by considering the following optimization function:

$$E = \sum_{i=1}^{N} (k_i - k_i^*)^2 + \varepsilon D$$

where N is the number of nodes, and k^* is the target degree of a single node we can easily obtain different network topologies. As an example, if we assume that to each node is assigned its own target degree k* according to the truncated scale-free distribution:

$$p(k_*) = \frac{k_*^{-\gamma}}{\sum_{k_*=k_{\min}}^{k_{\max}} k_*^{-\gamma}}$$

the resulting network wireless multi-hop network present a scale-free topology [28]. A similar approach was also proposed in [30], where it was shown that by applying a fully distributed and de-synchronized preferential attachment rule it was possible to generate a scale-free wireless multi-hop network. This is possible through a preliminary negotiation phase that is needed in order to progressively add new nodes in the network and, at the same time, maintain the targeted degree distribution.

7.3.1. Delay Tolerant Networks

Network connectivity is generally modeled by means of a connectivity graph, which incorporates existing connections among the different nodes of the network. A connectivity matrix then reflects the properties of the connectivity graph in a more concise form. However, such concepts vanish in the case of DTNs, where no traditional continually connected network can be observed. In fact, DTNs are characterized by an extreme dynamism originating from (i) the extreme sparseness (ii) the mobility of network nodes. In order to cope with such a time-varying environment it is possible to define a time-dependent connectivity matrix.

Let us consider first the case of a wireless network in which all nodes are static. In this case, we can build a matrix A, that we call the *connectivity matrix*, by looking at the fact that two nodes are within mutual communication distance. In other words, the (i,j)-th entry is positive, $A(i,j) > 0$ if and only if nodes i and j are within mutual communication range. For example, $A(i,j)$ could be the rate at which two nodes are able to transmit data to each other. In this way we could account for SNR variations while still being able to define a matrix A representing the "topology" of the network. Things drastically change when we consider a mobile network. In this case, indeed, there is no standard notion of network topology we can rely on for building the connectivity matrix. This task gets even more challenging if we focus on networks operated in the *subconnectivity regime*, in which no giant component exists.

The basic idea which we propose for building the connectivity matrix A is to consider an integrated version of the instantaneous network connectivity. Given a constant T, we construct the T-tolerant connectivity matrix A_T by considering all the meetings taking place over a time window of length T. The (i,j)-th entry $A_{T(i,j)}$ equals the number of meetings taking place in the time interval $[t_0,t_0+T)$, where t_0 is a given initial time instant. Such process is briefly depicted in Figure 6, where is it shown how the connectivity matrix is derived starting from the meetings pattern.

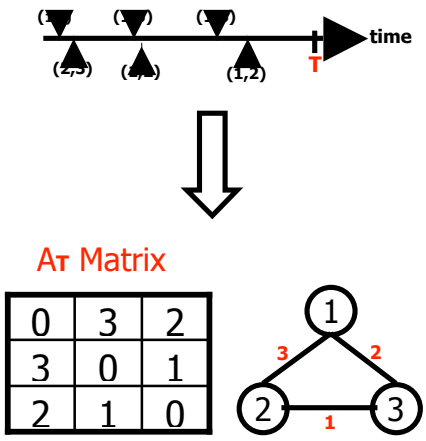


Figure 6: Connectivity matrix over time.

The definition of the *T*-tolerant connectivity matrix is worth some comments. In particular, the choice of an adequate time window *T* appears non-trivial. The value of *T* shall be consistent with the nodes' dynamics, so that the matrix A_T gives rise to non-trivial results. If *T* is too small, most entries of A_T would turn out to be zero, the network would show low connectivity and, hence the EVC analysis would not carry any significant information. The parameter *T* shall be understood as a kind of time constant of the system.

Starting from this definition of connectivity matrix in the case of DTNs, it becomes now possible to analyze some of the topological properties of the networks, and verify its robustness with respect to the system requirements imposed by the autopoietic P2P architecture.

As an example, in [94] such analysis was conducted and applied to the case of real-world mobility patterns, in which the meetings time sequence was obtained from real measurements [26]. Such contact traces were collected on a daily basis, and the core network is represented by 20 devices, using a IEEE 802.11 interface and mounted on buses. The connectivity properties (i.e., degree distribution, clustering coefficient, etc.) were evaluated starting from the definition of time varying connectivity matrix.

In Figure 7 and Figure 8, the degree distribution and network diameter are presented in the case of different window lengths. Results are then compared with the case of an Erdos-Renyi model.

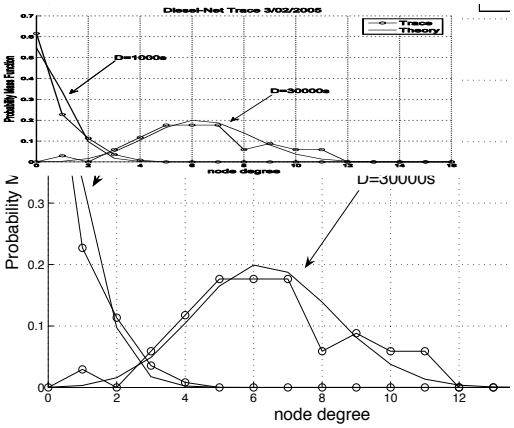


Figure 7: Degree distribution of connectivity graphs and comparison with the ER model, DieselNet trace 3022005; window length T = 1000; 30000 s.

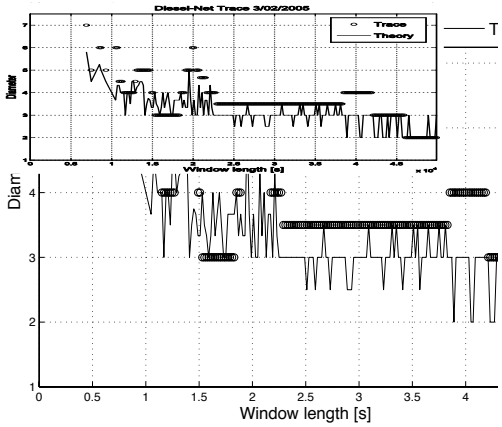


Figure 8: Diameter versus window length, DieselNet trace 3022005.

From such experimental results it is possible to observe that there is a significant match between the ER model and the resulting network connectivity. Hence, the structure emerging from a real common

connectivity pattern, such as the one of a network of busses, is far away from the one required by an autopoietic network infrastructure.

7.3.2. Supporting Autopoietic P2P Networks in DTNs

In the case of DTNs, supporting an autopoietic P2P network infrastructure is an extremely challenging task, since it is not possible to add/remove links between nodes where needed. In DTNs, the possibility of duplicating/removing links is related to the availability of *contacts opportunity*: the more are the contacts, the better (in a dynamic sense) the network is connected. However, such connectivity is constrained by the availability of mobile carriers capable of enforcing some critical links, or from the possibility of dynamically influencing the mobility pattern of some mobile nodes.

An opportunity to be explored for achieving better connectivity is the utilization of *throwboxes* [27], which are wireless nodes *thrown* in the environment to create a greater number of contact opportunities in DTNs. If an a-priori knowledge of the network is available, such special nodes can be appropriately placed in order to enforce a particular connectivity pattern among mobile nodes. Such throwboxes act as *hubs* of the mobile network, and would represent nodes with an extremely highly degree of connectivity.

7.4- Closing Remarks

Supporting distributed transactions over a mobile network results in an extremely demanding task. In this contribution, we have considered three distinct mobile network architectures: infrastructure-based networks, mobile ad-hoc networks, and delay tolerant networks. Each scenario results to be characterized by an increased *hostility* of the operating environment.

The first case refers mostly to cellular networks, where it is (almost) always possible to access a backend link for connecting to remote services. In this case, the major issue is how to efficiently deal with temporal disconnections from the fixed network and how to efficiently balance the load so to limit the number of blocking operations, and avoid unnecessary aborts. However, in this case the network connectivity is imposed by the base stations deployed by mobile operators. In order to achieve a more robust network infrastructure, thus enabling an autopoietic P2P network, it is necessary to introduce additional infrastructure. As an example, we have reported the case of a cellular network being combined with a MANET. In this case it becomes possible to obtain a scale free topology at the cost of additional complexity and hardware components being added to the system.

The second case considered is that of mobile ad-hoc networks. In this case, another degree of complexity is added by the absence of a fixed infrastructure, and by the increased dynamicity of the network nodes. In this case, nodes are expected to self-organize and dynamically adapt to the changing operating conditions. Also in this case, network connectivity is highly influenced by the nodes physical position. It is possible to dynamically vary the communication range of nodes, and obtain different network topologies. By choosing an appropriate “attachment rule” for the nodes, it becomes possible to obtain a scale-free network topology. However, it is also noticed that this, while being possible in theory, in practice is highly influenced by physical constraints (interference, obstacles, etc.) which make such topology constructions not always realistic.

As a last case, we have considered Delay Tolerant Networks where the nodes’ physical mobility is exploited to achieve network-wide delivery of messages. In this case, nodes contacts drive the communication process. We have described a way to model the connectivity properties of such networks and shown some experimental results obtained over real-world mobility patterns. Preliminary results highlight that the degree distribution is in this case similar to an ER model, and therefore not as robust as requested for distributed transactions. In this case, since the connectivity is defined by contacts opportunities, the only means to achieve a more stable and robust network topology is to increase the

number of contact opportunities. This can be achieved by means of additional relaying nodes, called throwboxes, or by partially influencing the mobility pattern of network nodes.

7.5- References

[1] K. Fall, “A delay-tolerant network architecture for challenged Internets”, in *Proc. of ACM SIGCOMM*, Karlsruhe, DE, 2003.
[2] S. Jain, K. Fall, and R. Patra, “Routing in a delay tolerant network”, in *Proc. of ACM SIGCOMM*, Portland, OR, 2004.
[3] . Chaintreau, P. Jui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, “Impact of human mobility on the design of opportunistic forwarding algorithms”, in *Proc. of IEEE INFOCOM*, Barcelona, ES, 2006.
[4] M. M. Bin Tariq, M. Ammar, and E. Zegura, “Message ferry route design for sparse ad hoc networks with mobile nodes”, in *Proc. of ACM MobiHoc*, Florence, Italy, 2006.
[5] A. Khelil, C. Becker, J. Tian, and K. Rothermel, “An epidemic model for information diffusion in manets”, in *Proc. of ACM MSWiM*, 2002.
[6] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H.Weiss, “Delay-tolerant network architecture”, 2005, IETF Internet Draft.
[7] S. Burleigh, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, “Delay-tolerant networking: an approach to interplanetary Internet,” *IEEE Comm. Mag.*, vol. 41, no. 6, pp. 128–136, Jun. 2003.
[8] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, “MaxProp:routing for vehicle-based disruption-tolerant networks,” in *Proc. of IEEE INFOCOM*, Barcelona, ES, 2006.
[9] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, “MobEyes: smart mobs for urban monitoring with vehicular sensor networks,” UCLA CSD, Tech. Rep. 060015, 2006.
[10] I. Chlamtac, M. Conti, and J. Liu, “Mobile ad hoc networking: Imperatives and challenges,” *Ad-Hoc Networks Journal*, vol. 1, pp. 13–64, Jul.
[11] E. Brewer, M. Demmer, B. Du, M. Ho, M. Kam, S. Nedevschi, J. Pal, R. Patra, S. Surana, “The Case for Technology in Developing Regions”, IEEE Computer Mag., vol. 38, no 6, pp.25-38, June 2005
[12] P. Mahonen, D. Trossen, D. Papadimitriou, G. Polyzos, D. Kennedy, “EIFFELTHE FUTURE NETWORKED SOCIETY A white paper from the EIFFEL Think-Tank”, December 2006
[13] Chor Min Tan; Chin Chin Wong, “Mobile Broadband Race: Friend or Foe?” In Proc. of. ICMB, Copenhagen, Denmark, June 2006
[14] P. Gupta and P. R. Kumar, “The capacity of wireless networks,” IEEE Trans. on Inf. Th., vol. 46, no. 2, pp. 388–404, Mar. 2000.
[15] K. Lyytinen, Y. Yoo, “Research Commentary:The Next Wave of Nomadic Computing” Information System Research, vol. 13, no. 4, pp. 377–388, Dec. 2002
[16] R. Ramanathan, J. Redi,.”A brief overview of ad hoc networks: challenges and directions”, IEEE Comm. Mag., vol. 40, no.5, pp.20-22, May 2002
[17] Panos K. Chrysanthis, "Transaction Processing in Mobile Computing Environment", In Proc. of IEEE Workshop on Advances in Parallel and Distributed Systems, 1993, Princeton, New Jersey, US
[18] Theodore Rappaport, “Wireless Communications”, Prentice Hall, 2002
[19] Jim Gray, Andreas Reuter, “Transaction Processing: Concepts and Techniques”, Morgan Kaufmann Publishers Inc., 1992
[20] Lee, M. and Helal, S. 2002. HiCoMo: High Commit Mobile Transactions. Distrib. Parallel Databases 11, 1 (Jan. 2002), 73-92.
[21] Dunham, M. H., Helal, A., and Balakrishnan, S. 1997. A mobile transaction model that captures both the data and movement behavior. Mob. Netw. Appl. 2, 2 (Oct. 1997), 149-162
[22] J Böse, S Böttcher, K Hahn, M Scholz, H Schweppe, “A Probabilistic Model for Transaction Persistency in MANETs”- In Proc. of MMS, 2006
[23] Serrano-Alvarado, P., Roncancio, C., and Adiba, M. 2004. A Survey of Mobile Transactions. Distrib. Parallel Databases 16, 2 (Sep. 2004), 193-230
[24] F. De Pellegrini, D. Miorandi, I. Carreras and I. Chlamtac, "A Graph-Based Model for Disconnected Ad Hoc Networks", in Proc. of IEEE INFOCOM 07
[25] K. Fall, W. Hong, and S. Madden. Custody Transfer for Reliable Delivery in Delay Tolerant Networks. Technical Report IRB-TR-03-030, Intel Research Berkeley, 2003.

[26] The disruption tolerant networking project at UMass. [Online]. Available: http://prisms.cs.umass.edu/diesel/
[27] An Energy-Efficient Architecture for DTN Throwboxes. by Nilanjan Banerjee, Mark D. Corner, and Brian Neil Levine. In Proc. of IEEE Infocom, Anchorage, Alaska, May 2007.
[28] W.Krause, I.Glauche, R.Sollacher, and M.Greiner. Impact of network structure on the capacity of wireless multihop ad hoc communication. Physica A, 338(3):633-658, 2004.
[29] Christian Bettstetter. On the Minimum Node Degree and Connectivity of a Wireless Multihop Network. In Proc. ACM Intern. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc), Lausanne, Switzerland, pp. 80-91, June 9-11, 2002.
[30] F Saffre;H Jovanovic1;C Hoile;S Nicolas. Scale-Free Topology for Pervasive Networks. In BT Technology Journal, Vol. 22, n. 3, pp 200-208, July 2004
[31] H.-Y. Hsieh and R. Sivakumar, "A hybrid network model for cellular wireless packet data networks". In IEEE GLOBECOM, November 2002.
[32] 3GPP TSR-RAN, "Opportunity Driven Multiple Access," 3GPP Technical Report, 3G TR 25.924, v1.0.0, Dec. 1999.
[33] Y. D. Lin and Y. C. Hsu, "Multihop Cellular: A new Architecture for Wireless Communications" in Proc. of IEEE INFOCOM. Telaviv.Israel, Mar. 2000.
[34] T. Harrold and A. Nix, "Capacity Enhancement Using Intelligent Relaying fro Future Personal Communication Systems," in the proceedings of IEEE VTC Fall, Boston, MA, USA, Sept. 2000
[35] C. Qiao and H. Wu, "iCAR: An Intelligwent Cellular and Ad Hoc Relay Systems," in Proceedings of IC3N, Las Vegas, NV USA, Oct 2000.
[36] G. Aggeleu and R. Tafazolli, "On the Relaying capacity of Next-Genaration GSM Cellular Network," IEEE personal Communications Magazine, vol. 8, no. 1, pp. 44-47, Feb 2001.
[37] J. Broeh et al, "A Pedomance Comparison of Multi-Hop Wireless Ad Hac Network Routing Pmtocols," in Proceedings of ACM MOBICOM, Dallas, TX USA, Oct. 1998.
[38] H. Luo and S. Lu, "A Topology-Independent Fair Queueing Model in Ad Hoc Wireless Networks," in Proceedings of IEEE ICNP, Osaka, Japan, Nov. 2000.
[39] H.-Y. Hsieh and R. Sivakumar, "On Using the Ad-hoc Network Model in Cellular Packet Data Networks,' in Proceedings of ACM MOBIHOC. Lausanne, Switzerland, June 2002.
[40] H. Wu et al., "Integrated Cellular and Ad-Hoc Relay Systems," IEEE JSAC, vol.19 no. 10, Oct 2001, pp. 2105-15.
[41] E. Yanmaz and O. K. Tonguz, "Dynamic Loan Balancing and Sharing Performance of Integrated Wireless Networks," IEEE JSAC, vol. 22, no. 5, June 2004, pp. 862-72.
[42] Sudhir Dixit, Evs, en Yanmaz, and Ozan K. Tonguz, "On the Design of Self-Organized Cellular Wireless Networks" IEEE Communications Magazine, vol. 43, no. 7, July 2005, pp. 86-93.
[43] Nuno Santos and Paulo Ferreira, "Making Distributed Transactions Resilient to Intermittent Network Connections", in Proc. of WoWMoM, 2006
[44] N. Santos, L. Veiga, and P. Ferreira, "Transactions policies for mobile networks", in Proc. of the 5th IEEE Int. Workshop on Policies for Distributed Systems and Networks, 2004

8. A Scale Free Network Algorithm and Simulation Environment (TI: Mikala P. Streeter & Jesús E. Gabaldón)

8.1- I. Introduction

With the latest boom of such Internet communities as Skype (the VOIP network) and social networks such as myspace.com and facebook.com, there is an increasing demand for an efficient way to maintain these networks as they grow and expand. There is also a need to design new networks without being able to predict how large the network will eventually become.

Scale free networks can meet this demand. By utilizing highly connected nodes in the network, the scale free design is able to ensure that the network does not lose its connectedness even as nodes fail at random.

We have designed an algorithm that ensures the network will follow the expected scale free pattern, as described by the equation f(x)=a*x^-k, given constant values for a and k. We simulated this algorithm using Java, creating a network of 1,000 nodes, each with a maximum degree of 300. The results of the simulation showed in the resulting log-log plot that the algorithm accurately models a scale free network.

8.2- Overview

8.2.1. *A.* The Components of the Network

1. Super node:
   - keeps track of all of the nodes in the network
     1. maintains a list of which other nodes every node in the network is connected to (called "Network Connections List")
     2. maintains a dynamic queue of the nodes in the network. A nodes position in the queue is determined by a nodes number of potential connections and its number of current connections (called "Node Queue")
     3. maintains a dynamic mapping of the different values of node degree in the network, from zero to the maximum value, to the nodes that have that number of connections. For example, if there are three nodes that have 4 connections, this map lists the three nodes as tied to node degree four. (called "Degree Map")
     4. maintains a dynamic list of the node degree levels in the network that are currently at their maximum capacity, as determined by the equation f(x)=a*x^-k (called “Bottlenecks List”)
   - allows nodes to join and leave the network, and to establish and break connections with other nodes
   - ensures that:
     1. No node is connected to itself
     2. Each node has only one direct connection to any another node
     3. There are no duplicate nodes in the network (determined by ensuring that no two nodes have the same IP and port)
   - stores the values for a & k, the variables in the equation f(x)=a*x^-k
2. Network node:
   - Contacts a known Super node to join the network, connect with other nodes, disconnect from other nodes, and exit the network
   - Maintains a list of the nodes to which it is connected (called "Node Connections List")
   - Regularly updates its number of remaining connections and Node Connections List
   - Maintains a boolean variable on whether it is currently trying to exit the network

8.2.2. B. The Parameters

1) a = the coefficient in f(x)=a*x^-k
2) k = the exponent in f(x)=a*x^-k
5) IP and port of each node

8.3- Protocol

8.3.1. A. To join network

When a new node wants to join the network, it contacts the known super node with a request to join the network. The super node tries to connect the new node to the first node in the queue, using the protocol described in section III.B. to connect the nodes. If it is not possible to connect the two, either because there is no other node in the queue or all the nodes in the queue are maximally connected, then the new node is added to the network without being connected to another node.

If, after the super node tries to connect the new node to all other nodes in the network, no connection has been made, the new node is allowed to join the network without any connection. The Node Queue and Degree Map maintained by the super node are re-evaluated to include this new node. But, the new node has to wait until another node requests to join the network before it can be connected to another node.

However, if a connection is made, the Degree Map is updated with the new node degree levels for both nodes. Both nodes are then notified of their new connection. They then both update their list of connections to include the other node and they also update their number of remaining connections to be one less than it was previously.

8.3.2. B. To establish connection

When two nodes already in the network want to make a connection, they must contact the super node and request that the connection be established.

Once the super node receives the request, it first checks to see if both nodes have at least one remaining connection and are not already connected. If both of those conditions are met, then the super node looks at the Degree Map and calculates the value for the maximum number of nodes for the node degree level that both nodes would join if the connection is made. This maximum number is calculated by finding the value f(x) (which equals a*x^-k), where x is the new node degree level for both nodes and a & k were pre-determined.

This maximum number is then compared against the number of nodes already at that node degree level. If the current number of nodes has not reached the maximum value at either level, then the connection is made.

All of the levels that are found to have reached their maximum capacity are added to the Bottlenecks List. When a new node tries to join the network, this list is used to re-connect a portion of the nodes at each level in the list, so that the network maintains an even distribution, according to the equation f(x)= a*x^-k.

If the maximum value has been reached at only one of the levels, then the super node checks to see if the maximized level is the level that will be left by the other node. In other words, the super node checks to see if the two nodes will be “switching” places in the maximized degree level. If so, then the connection is made. If not, the super node does not allow the connection, and notifies both nodes that the direct connection cannot be made.

If the maximum value has been reached at both levels, then the super node does not allow the connection. The super node notifies both nodes that the direct connection cannot be made.

If the maximum value has not been reached, the connection is made. Once a connection is made, the Degree Map is updated with the new node degree levels for all involved nodes. The Network Connection List and the Node Queue are updated as well.

8.3.3. C. To break connection

When two nodes already in the network want to break their connection, they must contact the super node and request that the connection be broken.

Once the super node receives the request, it first checks to see if both nodes are already connected. If not, it informs both nodes of this information.

If the nodes are connected, then the super node breaks the connection and informs the two nodes that the connection has been broken so that they can update their records. The super node updates its records as well.

If by breaking the connection, either node is left without any connections in the network and it is not trying to exit the network, then that node is connected to some other node in the network with remaining connections before being disconnected from the other node. If there is no other node in the network with remaining connections, the node without any connections in the network is pushed to the top of the node queue, so that it can be reconnected to another node as soon as possible.

8.3.4. D. To exit network

When a node wants to exit the network, it requests that the super node remove it from the network. The super node then goes through each connection between the node that wants to be removed and other nodes in the network, removing each one by one using the protocol defined in section III.C. above, until they are all removed. During this time, the exiting node is not allowed to be included in any new connections. Once all the connections are removed, the super node's records are updated to reflect the changes in the network.

If a node that was previously connected to this node is left without any connections in the network, then that node is immediately connected to some other node in the network with remaining connections. If there is no other node in the network with remaining connections, the node without any connections in the network is pushed to the top of the node queue, so that it can be reconnected to another node as soon as possible.

8.4- Simulation

8.4.1. A. Overview

Using Java, we performed a simulation of a scale-free network. We added 1,000 nodes to the network using the protocol described in section II. Each node was assigned a maximum node degree of 300 nodes, meaning it was possible for each node to have up to 300 connections. After all of the nodes were added, ten network connections were chosen at random to be disconnected.

The parameters a and k of the equation f(x)=a*x^-k were set to be 420 and 1.5, respectively. These values remained the same throughout the simulation. The IP and port of each node were assigned from one to 1,000.

8.4.2. B. Storing of Data

The results of the simulation were stored in four text files. The results of the inital addition of nodes to the network were saved in two files. One file stored information about the number of nodes at each node degree level after each time step, e.g. “There are X nodes with Y connections”.

973	23 5	46 1
1 409	22 3	44 2
2 128	25 4	45 1
3 76	24 5	48 2
4 46	27 4	53 1
5 32	26 4	58 1
6 25	29 4	68 1
7 23	28 4	84 1
8 19	31 3	98 1
9 17	30 4	126 1
10 14	34 3	288 1
11 13	35 3	292 1
12 10	32 3	293 1
13 10	33 3	294 1
14 9	38 1	295 1
15 8	39 2	296 1
17 7	36 2	297 1
16 8	37 3	298 1
19 6	42 4	299 1
18 6	43 1	300 1
21 5	40 7	
20 6	41 2	

Figure 1. This table shows an example of the data stored in the node information file. The first row in the first column is the number of the time step. At this time step, there are 973 nodes in the network. The subsequent rows detail the number of nodes at each degree level. For example, rows two and three of column one, say that there 409 nodes with one connection and 128 nodes with two connections.

The second file stored information about the state of the network after all of the nodes

had been added, i.e. which nodes are connected to which other nodes. This network information was stored in a triangular matrix, where 1's signified a connection and 0's signified the lack thereof.

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
```

Figure 2. This figure is one line of the matrix stored in the network information file. The 1s signify the nodes to which this node is connection. The 0s signify that this node is not connected to the nodes that correspond to those positions.

After the 10 randomly chosen connections were disconnected, this information (about the nodes and the network) was saved again in two new files to detail the state of the network after the disconnections were complete.

The data from these files was then used to plot a log-log graph of the number of nodes at each node degree level. The data was also analyzed using other software to further determine the effectiveness of the algorithm. The results are recorded in Section IV.

8.5- Results

Using a trial version of graphing software, we graphed the results of each run of the simulation – with the log of the node degree level on the x-axis and the log of the number of nodes at each degree level on the y-axis. In the early trials, there was a spike in the graph where the node degree level equalled one. This spike occurred because, as new nodes were being added to the network, they were being added to the same small group of nodes, and not being connected again after that. The nodes outside of this small group did not establish any more connections because they were unable to move up in the queue, which prioritizes nodes based on the total possible degree of each node and its current node degree. The current node queue degree continued to increase for those at the head of the queue, allowing them to maintain their position and ensuring that they would continue to be connected to new nodes joining the network until they had used all of their connections.

In order to force the node distribution to better follow the pattern described by the equation f(x)=a*x^-k, we added a function to remove these "bottlenecks" from the network. As mentioned in section I, the super node maintains a dynamic list of the node degree levels in the network that are currently at their maximum capacity (also known as the "Bottlenecks List"). When two nodes try to connect, if the super node finds that either (or both) of the levels that the two nodes will join after connecting has reached its maximum capacity, the super node adds that level to this dynamic list. The next time a node is added to the network, the super node checks this list and, if there are any entries in it, the super node goes through each node degree level corresponding to the entries in the list, connecting one-third of the nodes at each level and removing that entry from the list after adding the connections.

By making these new connections, the network is more evenly distributed because a portion of the nodes have changed degree levels, spreading the nodes throughout the graph, defined by the equation f(x)=a*x^-k.

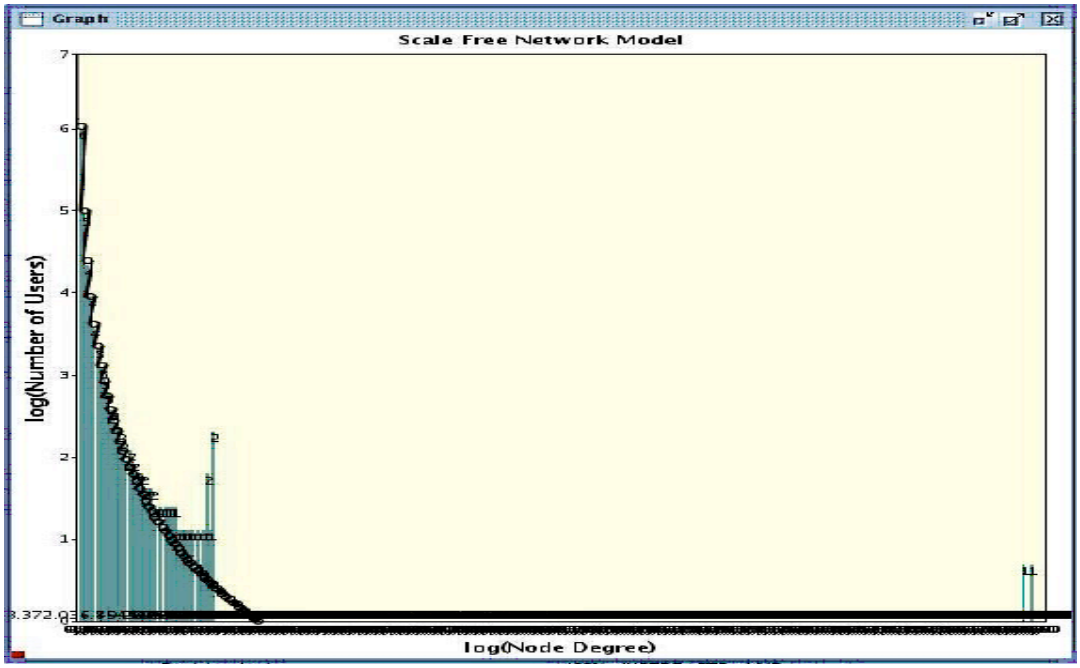


Figure 3. This graph is the result of adding the "bottleneck" dispersing functionality. It follows the pattern given by the equation for most of the graph. The few small spikes are minor in comparison to the height of the spikes before this functionality was added.

9. Conclusions

This has been a very wide-ranging report, and it may be helpful just to summarise the journey from beginning to end. However, let us recall first the primary motivation behind all this work.

We are at a critical point in the evolution of the Digital Economy. Sites for e-Commerce and Social Networking are facilitating significant changes to the way people do business and engage with each other. But this ecosystem is dominated by a small number of big players: Amazon; Expedia; Google; iTunes; FaceBook; and so forth. Not only does this bias this distribution of "species" in the ecosystem towards a small number of "Keynote" species, with threats to the pool of innovation (SMEs) that typically provide long-term stability in any healthy ecosystem. But they also provide barriers to adoption (the cost to a provider of delivering their service through Expedia, for example, is too high for small family run hotels), and threats to personal or local autonomy (contributors to social networking sites, for example, provide to the host free information about themselves, their friends and family, their holiday patterns, purchasing behaviour and more).

In contrast, this document provides an overview of an architecture for a digital infrastructure that is designed to break down these barriers to adoption and reinvigorate the generative capability of the Internet and World Wide Web. We have a model for supporting long-term transactions and collaborations between members of a community, without threatening the local autonomy of the participants in that community (Chapter 3). Work is underway to enable transactions to be dynamically generated from an open pool of resources, in order to meet not only expected, but also novel user requests (Chapter 4). These emerging local networks of collaboration can then be exploited to evolve a coherent digital infrastructure that efficiently supports the continued growth of the ecosystem (Chapter 5). Most importantly, this infrastructure is self maintaining, using only resources from the community it supports, with no critical points of failure or control. This offers not only economic benefits, but also strong environmental benefits in its philosophy of *reusing* spare computing capacity, instead of relying on purpose built farm(s) of dedicated servers. Simulation work (Chapter 6) is enabling us to gain further understanding of the validity of the assumptions behind our work. As we move towards trials of this work, it is important to consider the impact of the increasing use in mobile networks as wireless broadband becomes increasingly pervasive. This was addressed in Chapter 7, raising a number of research questions that will need further attention. We conclude in Chapter 8 with some experiments on controlling the generation of scale-free networks.

The core architecture is now quite stable, and we are beginning the transition from simulations to real-world trials of this work. Some trials are being supported from within OPAALS, but we also have confirmed funding from BERR, EPSRC and KTA in the UK for a three-year project to pilot the OPAALS approach in the Wildlife Conservation domain. The UK TSB (Technology Strategy Board) in the UK is also working with us to identify domains for further trials. Interestingly, we are not only getting interest in setting up trials in regional development schemes in Europe, but also interest from large organisations who see this approach as providing significant opportunities for an adaptive infrastructure that better supports innovation.