FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Classification of Space Based Computing Systems

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Wirtschaftsingenieurwesen Informatik

eingereicht von

Marion Altschach
Matrikelnummer 0225184

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: A.o. Univ. Prof. Dr. Dipl.-Ing. Eva Kühn
Mitwirkung: Dipl.-Ing. Thomas Scheller

Wien, 31.03.2016

(Unterschrift Verfasserin) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Classification of Space Based Computing Systems

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Business Engineering and Computer Science

by

Marion Altschach
Registration Number 0225184

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: A.o. Univ. Prof. Dr. Dipl.-Ing. Eva Kühn
Assistance: Dipl.-Ing. Thomas Scheller

Vienna, 31.03.2016

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Marion Altschach
Attemsgasse 5/1/107 1220 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasserin)

Acknowledgements

Any attempt to list all the brilliant people who have helped me on my journey through university and to writing this thesis would be like trying to count the stars in the heavens.Yet among these stand four individuals whose profound impact deserves special acknowledgement and to whom I would like to dedicate this thesis.

First and foremost I offer my sincerest gratitude to my supervisor, A.o. Univ. Prof. Dr. Dipl.-Ing. Eva Kühn, who has supported me thoughout this thesis with her incredible patience and knowledge whilst allowing me the room to work in my own way. Without her encouragement and effort this thesis would not have been written or completed. One simply could not wish for a better or friendlier supervisor.

Special thanks to Dipl. Ing. Thomas Scheller for his support, guidance and helpful suggestions. His guidance has served me well and I owe him my heartfelt appreciation.

And my partner, Felipe Juan-Hofer, who has given me his unequivocal support throughout, as always, for which my mere expression of thanks likewise does not suffice. Above all, I would like to thank my mother, Renate Altschach, for her personal support and great patience at all times .

For any errors or inadequacies that may remain in this work, of course, the responsibility is entirely my own.

Abstract

Space based computing systems have seen an increase in popularity since their first appearance in the mid-eighties. Hence, it is no wonder that what started with Linda, which is commonly regarded as the forbearer of this concept, has led to over one hundred modified, extended, enhanced or even to completely new shared data space implementations until 2016. Unfortunately, little effort has been made in classifying and organizing the collection of space based computing systems with the result that it is hard to know what's out there and how one can use them for certain requirements. This thesis introduces the most important approaches in this field with the aim to analyze, classify and compare these shared data space implementations in the hope of making the state-of-the-art visible.

In order to present the various space based computing systems a top-down approach has been chosen, which will start with a high-level overview of shared data space implementations and then work it's way through to increasingly detailed classifications of space families and classifications after certain use cases and features.

The principal conclusion of this survey was that there is not one space based computing system which solves all the issues like availability, transactions, asynchrony, near-time event notification, scalability, or security equally well. But of course, it can be argued that knowing about the various strengths and options of the single systems can help to better focus on development and progress in order to come one step closer to the perfect "reference"space based computing system.

Kurzfassung

Space based computing Systems haben seit ihren Anfängen, Mitte der achtziger Jahre, stark an Popularität zugenommen. Daher ist es auch nicht verwunderlich, dass seit Linda, welches gemeinhin als die Mutter dieses Konzeptes gilt, mehr als hundert, veränderte, erweiterte oder sogar zu völlig neue Shared data space Implementierungen bis 2016 entstanden sind. Leider wurde bis dato wenig Aufwand in Klassifizierung und Organisation dieser Space based computing Systeme gesteckt, mit dem Ergebnis, dass man nicht wirklich weiß, welche Möglichkeiten es auf dem Markt gibt und wie diese effizient für gewisse Zwecke einzusetzen sind. Diese Diplomarbeit stellt die wichtigsten Ansätze auf diesem Gebiet vor, mit dem Ziel die unterschiedlichen Shared data space Implementierungen zu analysieren, zu klassifizieren und zu vergleichen, in der Hoffnung den jetzigen Stand der Technik sichtbarer zu machen.

Um die unterschiedlichen Space based Computing Systeme in geeigneter Form vorzustellen, wurde ein Top-down Ansatz gewählt, welcher mit einem high-level Überblick beginnt und sich dann nach und nach detaillierteren Klassifikation nach Space Familien und Eigenschaften widmet.

Die Resultate dieser Arbeit zeigen, dass es im Moment nicht ein alleiniges Space based computing System gibt, welches sämtliche Thematiken wie Verfügbarkeit, Transaktionen, Asynchronität, zeitnahe Event-Benachrichtigung, Skalierbarkeit, oder Sicherheit in ihrer Gesamtheit optimal abbildet. Man kann jedoch argumentieren, dass das Wissen über die unterschiedlichen Stärken und Möglichkeiten der einzelnen Systeme, dabei helfen kann besser auf die Entwicklung und den Fortschritt zu fokussieren, um langfristig auf das bestmögliche Space based computing “Referenz” System hinzuarbeiten.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Parallelism, is one way to solve problems fast. Actually, it is the way.[1] These wise words, by Nicholas Carriereo[2] and David Gelernter[3], can be seen as one piece of the wholy grail when it comes to the big number of challenges programmers face nowadays.
Parallel and Distributed Computing (PDC) penetrates most computing activities in 2016 - effecting not only enterprises and academia but also the common user who all of a sudden is dependent on parallel processing. Just imagine how much fun a multi-user game would be without parallal processing. The answer to that question is easy. No fun at all. We are strongly dependent on parallelism in order to fulfill our present and future needs, e.g. software for high performance computing.
The past has also shown us that distribution and parallelism brings advantages like a decrease in computing time, an increase in precision of computations, the ability to solve larger and more complex problems, the possibility to take advantage of non-local resources, cost savings and much more.
There are many ways to build working parallel programs and David Gelernter and Nicholas Carriero introduced one of them, which also represents the beginning of what we now refer to as space based computing: The coordination language called Linda [Gel85].
Since that moment in the mid-eighties space based computing implementations have seen an increase in popularity. Hence, it is no wonder that what started with Linda, which is commonly regarded as the forbearer of this concept, has led to over one hundred modified, extended, enhanced or even to completely new shared data space implementations until 2016. Unfortunately,

[1] [CG90] page 7.
[2] Nicholas Carriero's is a Senior Research Scientist of Computer Science at Yale and his research centers on system issues in the development and deployment of software tools for parallelism. Information found under `http://www.cs.yale.edu/people/carriero.html`.
[3] David Hillel Gelernter was born in March 1955 and is an artist, writer, and professor of computer science at Yale University. Information found under `http://en.wikipedia.org/wiki/David_Gelernter`.

the few surveys out there have always focused on comparing just a small selection of space based computing systems. As a result software developers and researchers often not really know what's out there and how they can use them for their own requirements. This thesis introduces approaches in this field with the aim to analyze, classify and compare these shared data space implementations in the hope of making the state-of-the-art visible. The goal is to draw a picture that is as complete as possible to the best of my knowledge.

1.2 Problem statement

Section 1.1 already made us aware of the problem that today we face over one hundred shared data space implementations which we have no real overview of. If you are interested in the basic concepts of shared data space implementations, Linda might be the best solution to start with whereas if you are interested in using space based implementations for commercial reasons Apache River[4], Blitz[5], GigaSpaces[6], JXTA Spaces [Li01] or TCP Linda[7]might be better suited for your needs. But there are also solutions coming from the scientific field which often have a scientific significance but are also designed for commercial use. C-Linda [Gel85], Corso[8], dRuby and Rinda [Sek09], TripleSpace Communication[9], TuCSoN [OZ98a] and XVSM [KRJ05] can be seen as such solutions, to name just a few of them. Finally, one should not forget all the implementations which tackle specific issues such as security, mobility or near-time-data distribution.
Figure 1.1 gives an overview about the different use cases showing that 45% of all space based computing solutions considered for this survey (103 in total) have stayed in the scientific field and are therefore only available to a relatively small group of potential users. Figure 1.1 also shows us that 27% of all space based computing solutions taken into account for this survey, are only represented through one or two papers and don't offer a package which could be used neither for private nor for commercial reasons. Nevertheless these solutions have to be taken into account because they often offer new and interesting approaches which address relevant problems such as security or mobility. Good representatives for this group are Encrypted Shared Data Spaces [RDD+08], Limbo [DWFB97], LinqSpace [Gel11] and Swarm Linda [GMT08]. With 15% space based computing solutions with a sheer commercial focus show that not all concepts and implementations manage to reach a wider public. 13% of all space computing solutions in this survey represent a combination of solutions which are used as well in the scientific field as on an industry level.
This first rough classification aims to show that we deal with an inhomogeneous set of space based computing implementations which are often hard to compare with each other. Not only because of the different amount of information provided but because of their different backgrounds and focuses.

[4] http://river.apache.org/doc/spec-index.html
[5] http://www.dancres.org/blitz
[6] http://www.gigaspaces.com/
[7] http://lindaspaces.com/downloads/evaluation.html
[8] http://www.complang.tuwien.ac.at/eva/researchpublications.html
[9] http://www.tripcom.org

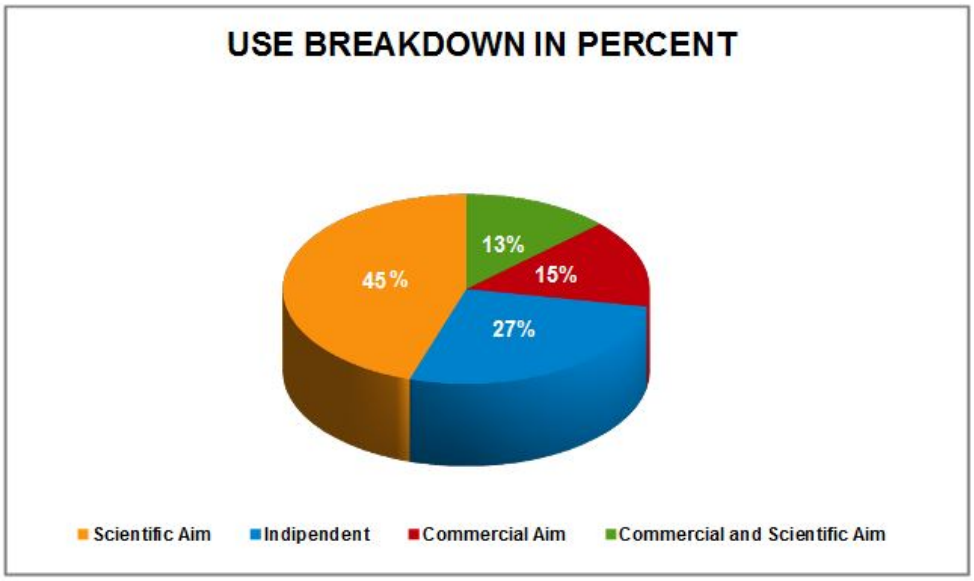


Figure 1.1: Use Breakdown in Percent

The main two challenges of this thesis can be described as follows:

- Find the right methodologies which allow bringing the diverse space based computing implementations into a relevant context with possible application scenarios and features of interest.
- Find the best visual models to present the huge amount of data gathered in an easy-to-understand way.

1.3 Aim of the work

The purpose of this thesis is to present a classification matrix of the 103 space computing systems that I have looked at. This matrix will give an overview of the most important approaches, present relevant features and use case scenarios. The main goal is to give software developers and researches a fast overview of what is out there and for what it can be used.

1.4 Methodological approach

The methodology chosen follows a top-down approach, which starts with an high-level overview of shared data space systems. After that the 103 space computing systems will be classified after families, application scenarios and different features.
Since a lot of the information has to be extracted from diverse literature and information from various websites, firstly it has to be processed in spreadsheets and then the findings will be presented through diagrams, visuals and tables.

1.5 Structure of work

This thesis is composed of five chapters and a comprehensive appendix. Chapter 1 gives an overview about this thesis including motivation, problem statement, aim, methodologies used and its structure.
Chapter 2 sets the scene for this thesis discussing fundamental terms and concepts in order to create a common understanding. Chapter 2 also includes a historical background and looks into the development of the space based computing paradigm over the course of time. Finally Linda, the forbearer of space based computing implementations is presented in detail along side with other interesting space based computing implementations.
Five application scenarios of interest will be described in chapter 3 before the actual classification process.
Chapter 4 focuses on the classification of space based computing implementations starting with classification in space families and application scenarios. After that the spaces will be classified into standard features and more special topics like transactions, notifications, persistency, scalability, security, remote access, life cycle management et cetera. One feature after the other is presented and then the space based implementations which adress this feature are presented and brought into system.
Chapter 5 summarizes the findings of the analysis and discusses where there is still room for adjustment, improvement and further development. As a consequence it makes sense to look at the latest research and developments of 2015 and 2016.
The appendix is structured into five sections:

- A glossary which presents all investigated space based computing systems in as much detail as necessary and gives information on literature.
- A list of figures.
- A list of tables.
- Acronyms.
- Bibliography.

CHAPTER 2

Fundamentals

2.1 Explanation of concepts and terms

Since a few terms and concepts are used repeatedly throughout this thesis it makes sense to explain them to reach a common understanding and offer the reader a quick reference if needed so.

2.1.1 Distributed systems

Today one can find distributed systems in various everyday services and a wide field of application domains as shown in figure 2.1[1]. The most popular one is probably the Internet. But also mobile phone networks, corporate networks, factory networks, campus networks, home networks et cetera [CDKB11] share characteristics which make them part of a family commonly known as distributed systems.
Literature shows that there is considerable disagreement about how to define a distributed system. Taking this into account it is helpful to define a few other things first in order to make the final definition of a distributed system easier to understand. [2]

- *A program is the code you write.*
- *A process is an instance of this program that is being executed.*
- *A program needs a processor in order to be executed.*
- *A message is used to communicate between processes.*
- *A network is the infrastructure that links computers, workstations, terminals, servers et cetera.*

[1] [CDKB11] page 4.
[2] `http://code.google.com/edu/parallel/dsd-tutorial.html`

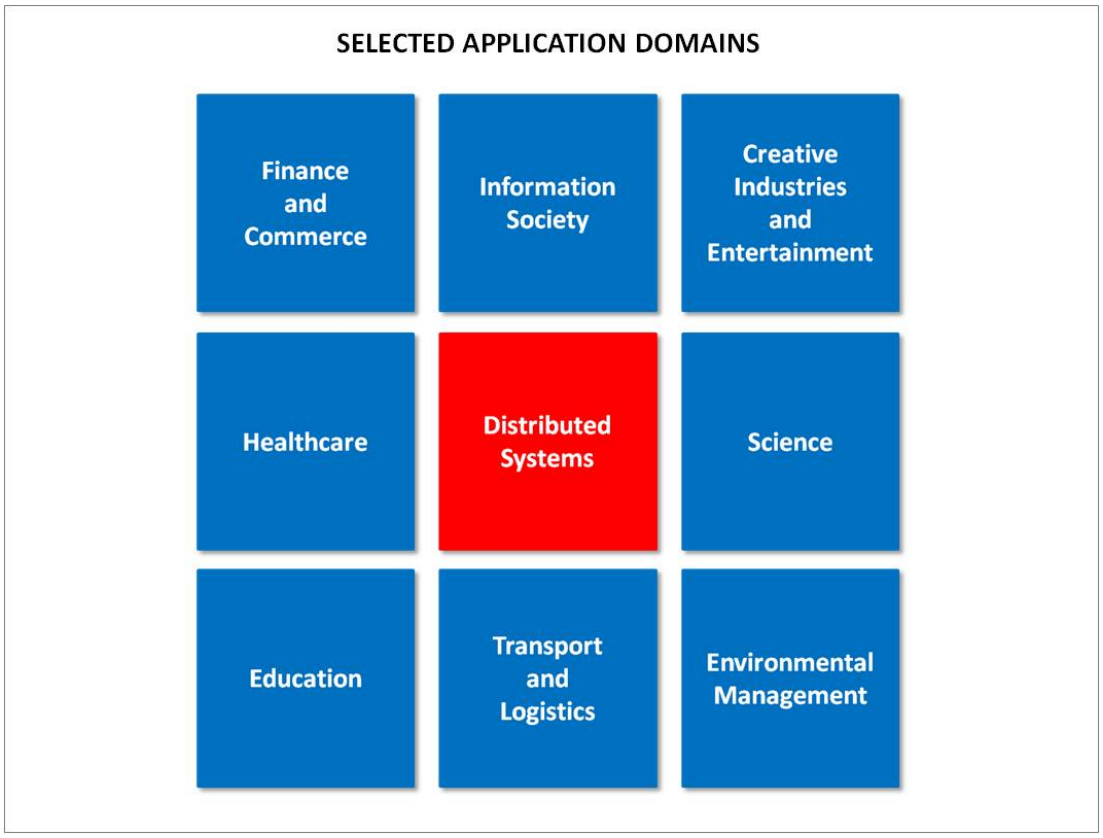


Figure 2.1: Selected Application Domains For Distributed Systems

Having defined these terms, we can now give a classical definition of a distributed system proposed by [BST89] that most people might agree on:
A distributed computing system consists of multiple autonomous processors that do not share primary memory, but cooperate by sending messages over a communications network.[3]
Two main advantages of distributed systems are obviously an improved performance through the use of parallelism – as already stated in the introduction - and an increased availability and reliability through the exploitation of redundancy.
But there are also challenges when constructing distributed systems like heterogeneity of their components, openness, security, scalability, failure handling, concurrency of components, transparency and providing quality of service.[4]
Modern distributed applications like multiplayer online games, financial trading systems or distributed multimedia systems [CDKB11] demand for new and effective solutions to address these challenges. In the next section an innovative and powerful concept is presented. Space based computing.

[3] [BST89] page 263.
[4] [CDKB11] page 1.

2.1.2 Space based computing

Chapter 2.1.1 already pointed out that we expect modern distributed systems to be reliable, scalable, simple and reasonably priced. But reality shows that distributed systems often are unreliable, not scalable, hard to manage and costly [Kue11].
The Space-Based Computing[5] paradigm, based on David Gelernter's publication Generative communication in Linda [Gel85], addresses this contradiction through decoupling interaction in the following three dimensions [Kue11]:

- Time: *Applications can read and write data whenever they want to.*[6]
- Space: A space is used as communication medium for all applications.
- Reference: Applications do not need to know each other in order to communicate.

The best way to define space based computing is to follow the definition given in the spacebasedcomputing.org manifest [EK07]:
Space based computing is based on the notion of a common, abstract space connecting distributed processing entities over a network. Instead of explicitly exchanging messages between individual processes or performing remote procedure calls, processes communicate and coordinate themselves by simply reading and writing distributed data structures in a shared space.[7]
A main advantage of space based computing lies in the high level abstraction it offers and therefore the application code can stay simple whilst a lot of functions are implemented by the space itself. This allows a shorter and less costly development for distributed applications.

2.1.3 Virtual shared memory paradigm

A virtual shared memory[8] *implements a shared-memory programming model in a distributed-memory environment.*[9] Once more Linda serves as an example because its tuple space can be seen as a virtual shared associative memory.
Diverse applications can interact with each other through this shared virtual data space. *The interaction is coordinated in a structured and secure way*[10] where finding the relevant data is possible without knowing its specific location.
[Kue98] states that although much research has been undertaken in this field, virtual shared memory solutions failed to reach a status where one would refer to them as standard.

[5] SBC
[6] [Kue11] page 5.
[7] [EK07] page 3.
[8] VSM
[9] [BS99] page 13.
[10] `http://www.complang.tuwien.ac.at/eva/SBC-Group/sbcGroupIndex.html`

2.2 Historical background

This section focuses on the historical background of space based computing implementations. In a first step it will be illustrated which spaces have been introduced over the course of time, how long they are respectively were active and when they had their peak-periods.
In a second step we bring the appearance of these space based computing implementations into context with certain industry needs and technological mile stones.

2.2.1 Historical evolution of space based computing implementations

Everything started in 1985 with a paper by David Gelernter and his colleagues from Yale University. The paper was called Generative communication in Linda [Gel85] and marked the beginning of a concept to which we now refer to as tuple spaces.
The Linda programming language *supports the concept of global object coordination*[11] and can be seen as the forbearer of all subsequent space based computing implementations and is therefore of particular interest. The main concept in Linda is that of a tuple space which can be seen as an a *logically shared associative memory*[12] that contains tuples and allows processes to communicate with each other by writing and reading or reading and removing these tuples in and out of the shared tuple space.
In the mid-eighties and early nineties the Linda model received a lot of initial interest [Wel05] and early successor implementations like BaLinda Lisp [YW90], Eiffel Linda [Jel90], Kernel Linda [Haz93], Klava [BDNP01] and Lana [BR02] followed inter alia.
From the moment Linda has been introduced, Gelernter has envisioned multiple tuple spaces to enhance Lindas possibilities. [Gel89]. Melinda[13] [Hup90] which was also developed at Yale focused as well on multiple tuple spaces and can be seen as the forbearer of the concept of multiple tuple spaces.
Although it became quieter about the concept of tuple spaces a lot has happened in the mid-nineties. The research at Yale has led to the establishment of the *Scientific Computing Associates, Inc.*[14] which in the late nineties has presented JParadise™ [Ass99], TCP Linda[15] and NetWorkSpaces™[16]. Also Bauhaus Linda [CGH97], Law-Governed Linda [ML95], PyLinda[17] and PyBrenda[18] have been presented amongst others in the mid-nineties.
In the late nineties the concept of the Linda tuple space was able to arouse the interest of companies which started to offer commercial implementations of Linda. This marked a milestone for the concept of tuple spaces because a wider public was reached and not only the scientific community. JavaSpaces was introduced as a component of the Jini[19] system by *Sun Mi-*

[11] `http://en.wikipedia.org/wiki/Tuple_space`
[12] [OG02] page 344.
[13] Linda with multiple tuple spaces
[14] `http://www.lindaspaces.com`
[15] `http://lindaspaces.com/downloads/evaluation.html`
[16] `http://www.lindaspaces.com/products/NWS_overview.html`
[17] `http://code.google.com/p/pylinda/`
[18] `http://wiki.python.org/moin/PyBrenda`
[19] `http://river.apache.org/`

crosystems[20] [Wel05] and was able to secure itself a place under the best known space based computing implementations. Other Java based implementations with an commercial aim presented in the late nineties were Autoevospaces by IntaMission, TSpaces [WMLF98] by IBM and GigaSpaces[21] by GigaSpaces Technologies with the latter of particular importance. In 2016 GigaSpaces offers a wide set of services and is a professional space-based computing implementation.
Also the scientific community actively developed new features for the existing implementations and also addressed new use cases mainly lying in the sector of mobility, security and the topic of garbage collection. Such space based computing systems are amongst others Blossom [GSW97], Lime [PMR99], Limone [FRH04], Swarm Linda [GMT08] and TucSon [OZ98a]. Especially TucSon shows that the scientific community tried not only to extend the classical Linda concept anymore but also to alter its ideas effectively. TucSon differs from Linda because it focuses on tuple centres (instead of tuple spaces), which allow to define and tailor the communication channel to the respective use case.
Between 2000 and the end 2015 we can see the presentation of quite a few systems, that are often presented through only one paper or a not well maintained website. Nevertheless, they often address niche use cases or rare features and therefore also contribute to the successful evolution of space based computing systems. Representatives for this group are Crudlet[22], D-Tuples [JXJY06], eLinda [Wel05], Glinda [KA07], LinqSpace [Gel11], PadSpace [LT09] and Tagged sets [OH05]. In the course of this thesisl refer to them as independent systems.
From 2005 on space based computing implementations often represent a hybrid form where innovation is driven in the scientific community but the focus lies as well on scientific topics as the commercial side. Good examples for such hybrid forms are Corso[23], dRuby and Rinda [Sek09], Klaim [DNFP98], TinyLime [CGG+05c], Triple Space Communication[24], TucSon [OZ98a], Xcoordination Application Space[25] and XVSM [KRJ05]. The Space Based Computing Group of the Vienna University of Technology is involved in the development of Corso, Triple Space Communication, Xcoordination Application Space and XVSM.
Figure 2.2 visualizes the distribution of space based computing implementations per aim from 1985 until the end of 2015. Due to space reasons the years 2012 to 2015 will be displayed in one column in all figures. In case specific information for this period is available it will be adressed in the text. The information on the figure underscores the evolution of space based computing implementations showing that space based computing implementations saw a continuous rise in popularity between 1999 and 2007. After 2008 many independent space based computing solutions have disappeared from the screen and the trend goes towards the improvement of already stable solutions like XVSM. This explains why in 2015 only twenty nine implementations are active whereas in 2007 and 2008 fourty solutions were active.
Implementations with a sheer scientific aim have been dominant between 2000 and 2008 but have been overtaken by implementations which combine a scientific and commercial aim in

[20] http://en.wikipedia.org/wiki/Sun_Microsystems
[21] http://www.gigaspaces.com/
[22] http://javaspaces.homestead.com/files/javaspaces.html
[23] http://www.complang.tuwien.ac.at/eva/researchpublications.html
[24] http://www.tripcom.org
[25] http://www.xcoordination.org/home

2015. Implementations with a commercial aim have started to enter the market in the late nineties and can be considered relatively stable since 2007 with ten implementations on the market. The independent space based computing systems always represented the minority.

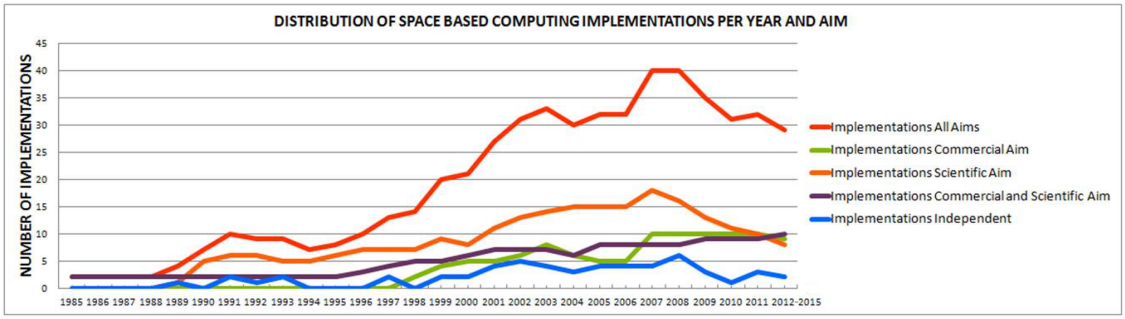


Figure 2.2: Distribution of Space Based Computing Implementations per Year and Aim

Figures 2.3 represents a detailed historical overview of space based computing implementations adding information on the respective aim of the implementations. Due to layout issues the legend for 2.3 underlies the same colour code as shown in figure 2.2. A big representation of this figure can be found in the appendix of this thesis under section B.
No information on the life span of HTML Page spaces [CR02], SemiSpaces [26] and SmallSpaces[27] could be found and is marked in both figures with the remark not available.
Both Linda and C-Linda are marked as active over the entire time horizon because although the original Linda implementations are certainly not state-of-the-art anymore, a lot of Linda-derived or at least influenced systems are still up and running in 2016.

[26] `http://www.theserverside.com/news/thread.tss?thread_id=55069`
[27] `http://en.wikipedia.org/wiki/Tuple_space`

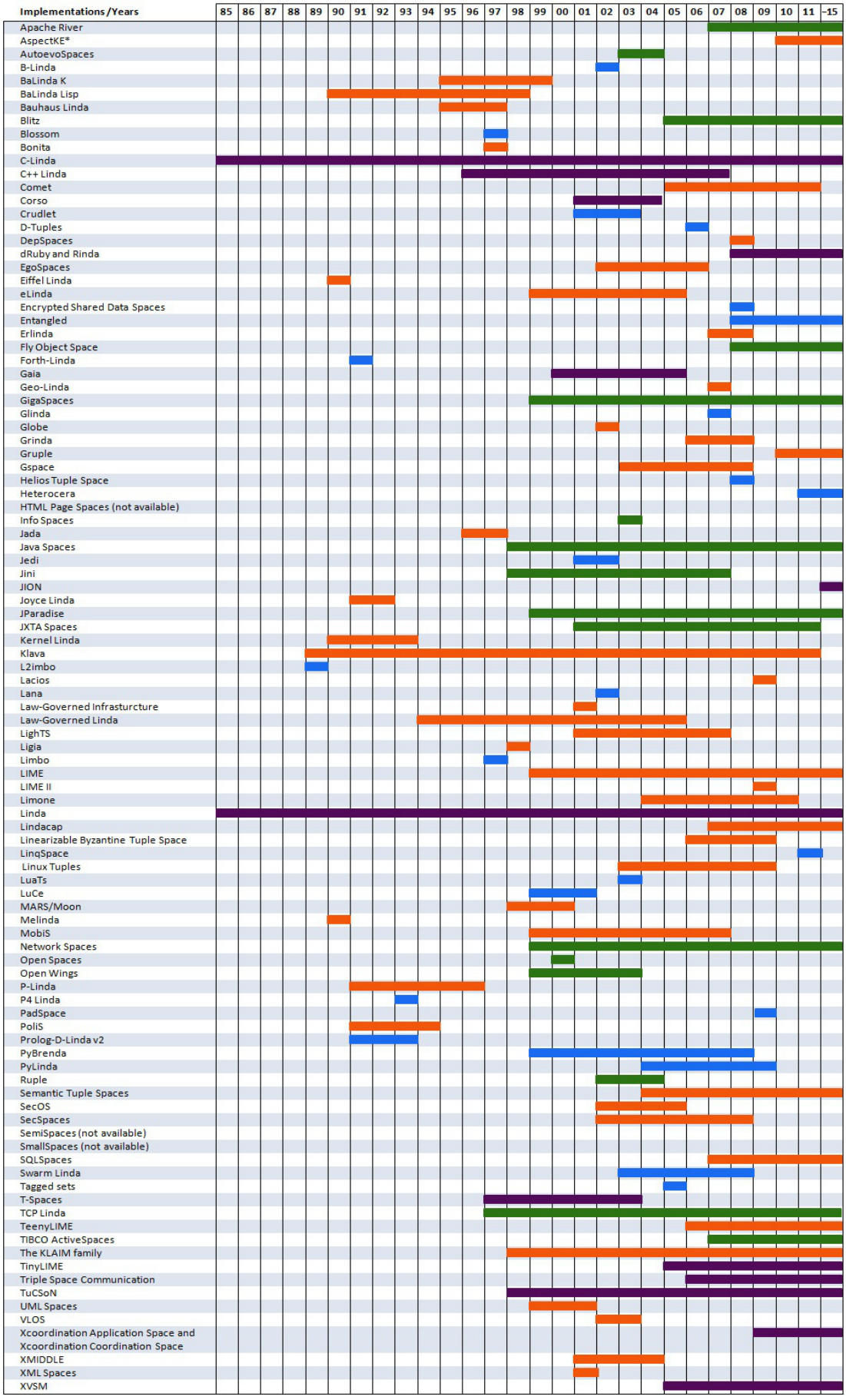


Figure 2.3: Detailed Historical Overview of Space Based Computing Implementations per Aim

2.2.2 Historical evolution of space based computing implementations in context

This section brings certain trends in the field of space based computing in context with technological milestones and industry needs.
Distributed computing has truly become a part of our everyday life and is undergoing continuous change since its appearance in the nineteen-sixties. Now in the 21st century we often use distributed systems in scenarios- e.g. online-shopping, online-banking or data sharing - *that depend crucially on security.*[28] For a very long time security was mostly of interest for the government and the military but when distributed systems started to get used by the industry in the mid-nineties a wider public started to be affected by this topic. Topics like cryptography, authentication and access control were discussed not only by the scientific community but also by the commercial world. Figure 2.4 shows that the topic of security reached the space based computing world with a little delay at the beginning of the twentyfirst century introducing implementations like SecOS [VBO03], SecSpaces [BGLZ02] VLOS [MCW02] and Law-Gowerned Linda [NBCdSFCL07].
In 2008 Encrypted Shared Data Spaces [RDD+08] and Lacios [ZBS09] were introduced at the peak of space based computing implementation focusing on security. Other implementations focus as well on security aspects but are more versatile than the solutions presented above. XVSM is such an example [KC12].

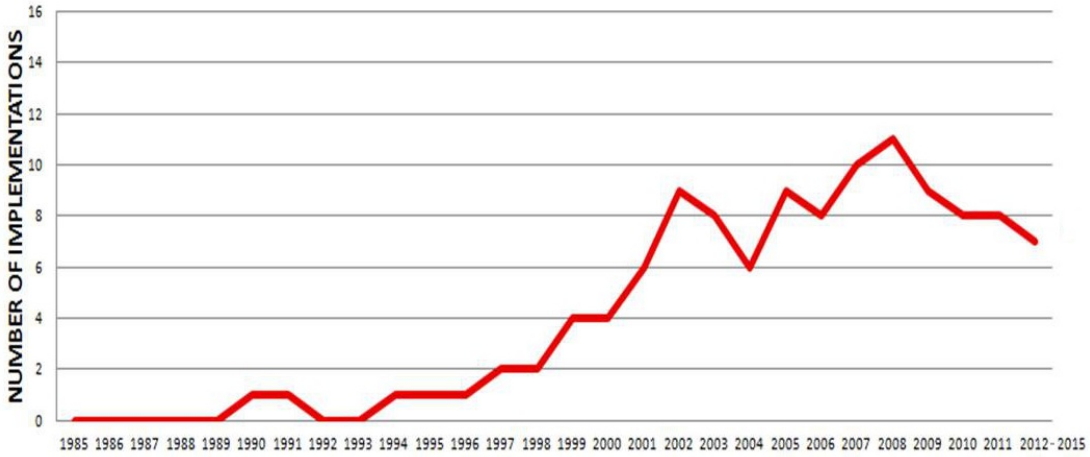


Figure 2.4: Distribution of SBC Implementations per Year Supporting Security

Scalability in distributed systems has always been a topic of big interest as well in the scientific community as in the industry. We call a distributed *system scalable if it can be economically deployed at a range of scales, in both small and large configurations.*[29] This might sound pretty easy but in fact it poses a few challenges[30]:

[28] [CDKB11] page 469.
[29] [JWM00] page 1.
[30] [CDKB11] page 20, ff.

- The costs of physical resources are not easy to control as the demand for a resource grows.
- A growing system often struggles to keep its performance at a constant level.
- The bigger the system the more likely it is that you have to deal with performance bottlenecks.
- The system and application software should not need to be changed when the scale of the system increases.

Figure 2.5 shows that the topic of scalability for space based computing implementations really took off in the mid-nineties and this can be brought in context with the rapid growth of the internet[31] during this time. The numbers of space based computing implementations dealing with scalability were steadily growing until 2007 and are now stabilizing because implementations are exiting the market. But nevertheless scalability is one of the most important topics when it comes to distributed systems and various tactics have been presented to deal with it in an effective way. At the moment further details on the different approaches will be neglected but chapter 4.12 will discuss them in detail.
Space based computing implementations addressing the topic of scalability are amongst others Apache River[32], BISSA [WWF+10], Comet [LP05], GigaSpaces[33], Globe [LS02], Java Spaces [PTW01], Linda [MTW01], OpenWings[34], Swarm Linda [GMT08], T-Spaces [WMLF98], Triple Space Communication [FKS+07], Xcoordination Application Space[35] and XVSM [Lwe10]. Looking at this enumeration of space based computing implementations one can come to the conclusion that implementations with a commercial aim have a strong incentive to deal effectively with the topic of scalability. 56 % of all commercial space based computing systems adress the issue of scalability, including Apache River, Fly Object Space, GigaSpaces, Java Spaces, JParadise, Network Spaces, Open Wings, TCP Linda and TIBCO ActiveSpaces.

[31] http://en.wikipedia.org/wiki/History_of_the_Internet
[32] http://river.apache.org/doc/spec-index.html
[33] http://www.gigaspaces.com/
[34] http://www.openwings.org
[35] urlhttp://xcoappspace.codeplex.com/

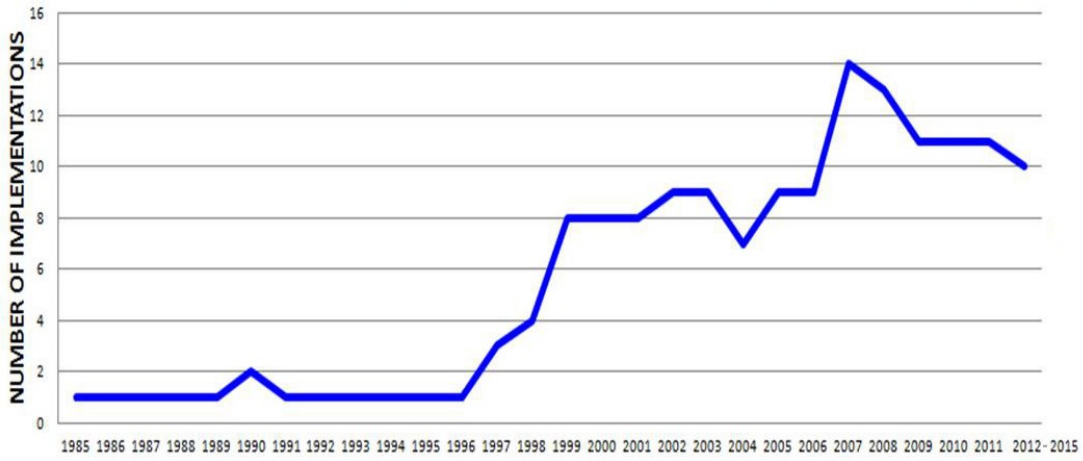


Figure 2.5: Distribution of SBC Implementations per Year supporting Scalability

Garbage collection is a critical problem for administratively decentralized distributed systems which manage distributed objects.[36]
A lot of research has been done in order to enable modern distributed systems to effectively deal with complex life cycle management tasks [PS95] which should not be handled by users in order to avoid errors. Hence, it is surprisingly that only a few space based computing implementations deal with this topic. One possible explanation for that might be that other topics like security or scalability seemed more relevant in the short term.
Figure 2.6 shows that the topic of life cycle management in space based computing implementations first appeared in the late nineties and since then only a few implementations are seriously dealing with this topic although it will be of great significance in the future when we have to manage even more data than we have to manage now. Representatives for such implementations are for example Ligia [MW98], Lindacap [UDI09], PyLinda[37] and XVSM [KRML08a].

[36] http://www.objs.com/workshops/ws9801/papers/paper015.html
[37] http://mail.python.org/pipermail/python-announce-list/2004-April/003054.htmland

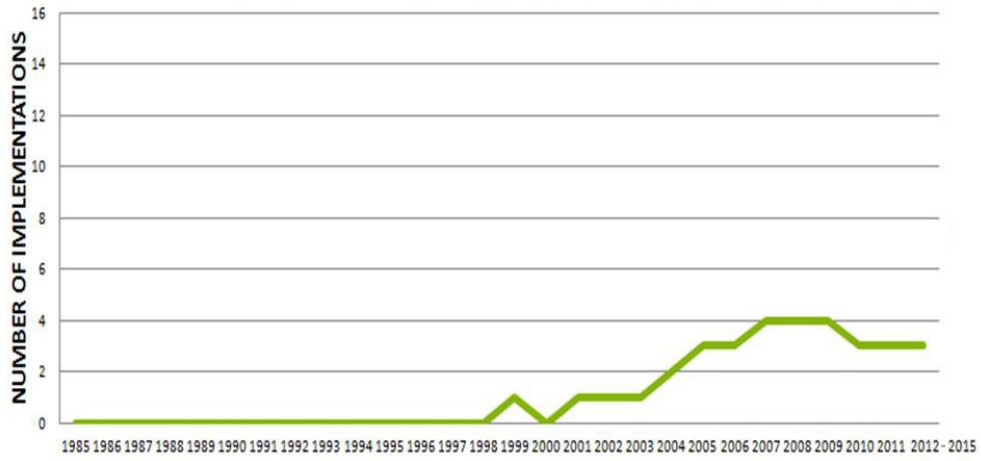


Figure 2.6: Distribution of SBC Implementations per Year supporting Life Cycle Management

Extensibility is not only a nice-to-have feature it is a must when a distributed system wants to be successful on the commercial market. Let's define extensibility in context with space based computing systems as the possibility of expanding or adding features to a system taking into account the level of effort required to implement such an extension. The second part of this defintion is of great importance since it prevents the assumption that just by changing the source code of a space based computing system it can be called extensible. What counts is the quality that features, modules or components can be used when needed and be switched off if not. Chapter 4.7 will discuss extensibility in more detail.
Extensibility in space based computing implementations has become a real topic in the late nineties where different use cases all off a sudden demanded different functionalities of the implementation. Figure 2.7 illustrates this trend which is still of importance in 2015. Commercial implementations like GigaSpaces and T-Spaces offer a certain degree of extensibility as well as implementations like Corso, Kernel-Linda [Haz93], LighTS [PB05], Semantic Tuple Spaces [TN04] or XVSM with the latter of particular importance for this topic.

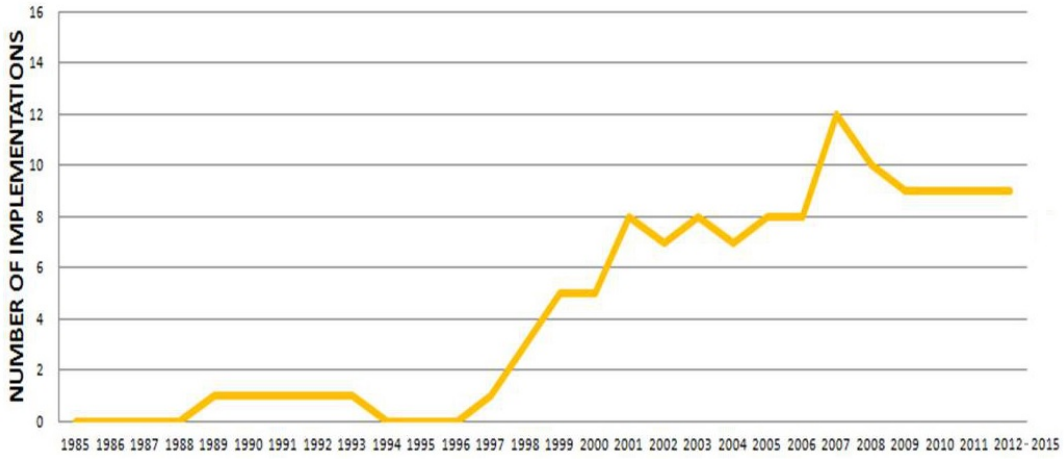


Figure 2.7: Distribution of SBC Implementations per Year Focusing on Extensibility

Now we have seen that in the nineteen-eighties the research of distributed system mainly focused on topics like remote communication, fault tolerance, high availability or security. In the nineteen-nineties a new trend called mobile computing entered the stage. Researchers started to deal with topics like mobile networking, mobile information access, support for adaptive applications and location sensitivity.
This development is clearly reflected in figure 2.8 which shows that the first space based computing implementations focusing on mobility were introduced in the late nineties as a reaction to trends like mobility, ubiquitous computing [Wei91] or hand-held computing. A few examples for such space based computing implementations are Jini[38] as well as the entire LIME family including LIME [PMR00], Teeny Lime [CMMP06], Tiny Lime [CGG+05c] or TOTA[39] [MZ04]. Figure 2.8 shows also that from the late nineties to 2008 the number of space based computing implementations supporting mobility saw a constant rise.

[38] `http://river.apache.org/`
[39] TOTA has not been included in the overall survey but will be discussed later in the closer context of mobility.

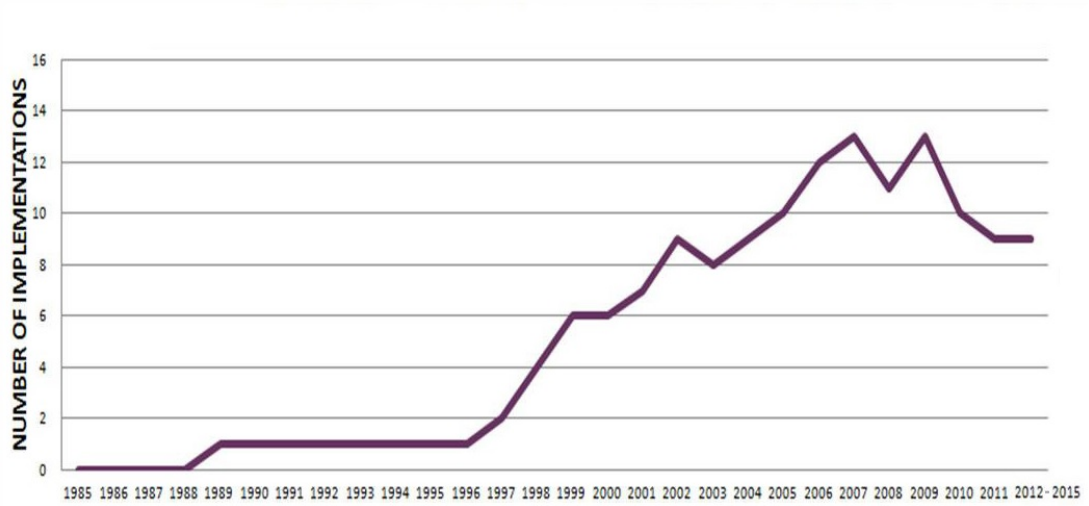


Figure 2.8: Distribution of SBC Implementations per Year supporting Mobility

In chapter 1.2 it was already stated that I found 103 space based computing systems when I started the thesis in 2013. On the other hand side over 2.500 documented programming languages exist[40]. Hence, it is no wonder that over the course of time a lot of programming languages integrated the ideas of Space Based Computing systems. Figure 2.9 shows the distribution of approaches combining the concept of Space Based Computing with programming languages over the course of time. Representatives are marked in figure 2.11.

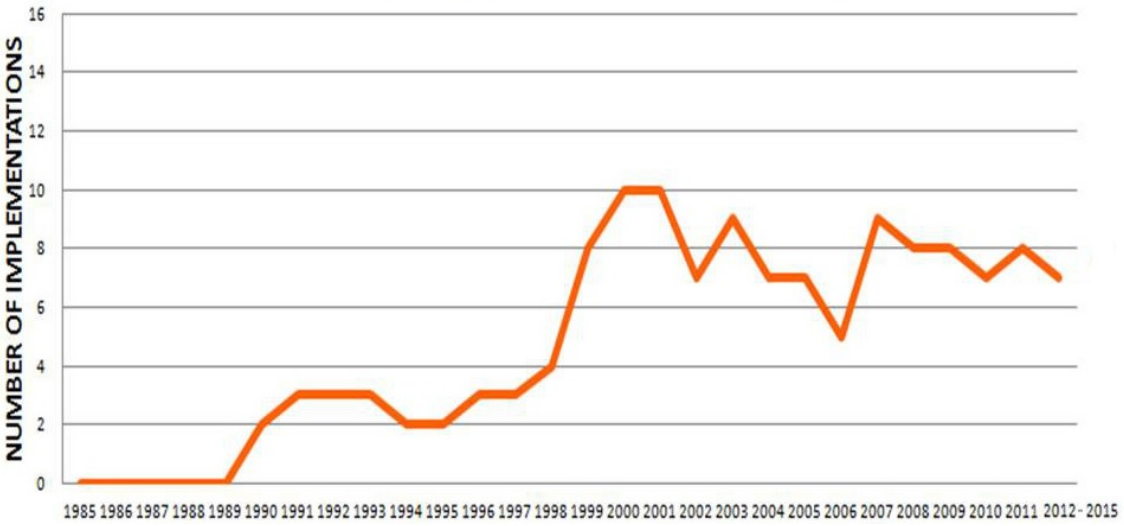


Figure 2.9: Distribution of SBC Implementations per Year supporting Programming Languages

[40] oreilly.com/news/graphics/prog_lang_poster.pdf

Figure 2.10 illustrates the different focus groups of space based computing implementations, which were discussed before separately, in comparison with each other. Looking just at the respective peaks what gets clear is that topics like scalability and mobility are the leading areas of interest followed by extensibility and security. Combining the ideas of Space Based Computing with different programming languages has been always of stable interest but does not seem as important as the focus groups before. The bottom of the table is represented by space based computing implementation focusing on life cycle management.

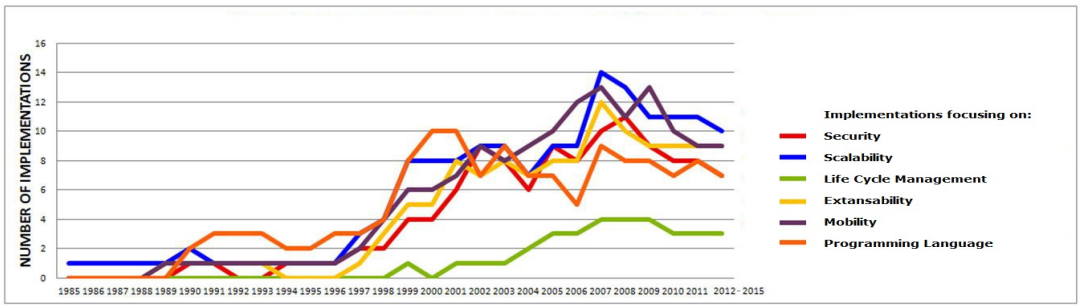


Figure 2.10: Distribution of SBC Implementations per Year and Focus Group

Figure 2.11 gives a detailed overview on what topics the particular implementations focus. Implementations can focus on more than one aspect.

Implementations /Focus Groups	Security	Scalability	Life Cycle Management	Extensibility	Mobility	Programming Language
Apache River	●	●		●	●	
AspectKE*	●					
AutoevoSpaces						
B-Linda						
BaLinda K						●
BaLinda Lisp						●
Bauhaus Linda						
Blitz						
Blossom	●	●				
Bonita						
C-Linda						
C++ Linda						●
Comet	●	●				
Corso	●		●			●
Crudlet						
D-Tuples						
DepSpaces	●	●				
dRuby and Rinda						●
EgoSpaces					●	
Eiffel Linda						●
eLinda		●				●
Encrypted Shared Data Spaces	●					
Entangled						●
Erlinda	●	●		●	●	●
Fly Object Space						
Forth-Linda						●
Gaia						
Geo-Linda						
GigaSpaces	●	●		●		
Glinda						●
Globe		●				
Grinda						
Gruple						●
Gspace				●		
Helios Tuple Space		●				
Heterocera						
HTML Page Spaces (not available)						
Info Spaces						
Jada						●
Java Spaces		●				●
Jedi						
Jini	●	●		●	●	
JION					●	
Joyce Linda	●					
JParadise		●				
JXTA Spaces						
Kernel Linda				●		
Klava					●	
L2imbo				●		
Lacios	●				●	
Lana	●				●	
Law-Governed Infrasturcture	●					
Law-Governed Linda	●					
LighTS				●		
Ligia			●			
Limbo					●	
LIME					●	
LIME II					●	
Limone					●	
Linda		●				
Lindacap			●			
Linearizable Byzantine Tuple Space	●				●	
LinqSpace						●
Linux Tuples						●
LuaTs						●
LuCe						●
MARS/Moon						●
Melinda		●				
MobiS		●			●	
Network Spaces						●
Open Spaces						●
Open Wings	●	●		●		●
P-Linda						
P4 Linda						
PadSpace						●
PoliS						●
Prolog-D-Linda v2						●
PyBrenda						●
PyLinda			●			
Semantic Tuple Spaces				●		●
Ruple						
SecOS	●					
SecSpaces	●					
SemiSpaces (not available)		●				
SmallSpaces (not available)						
SQLSpaces				●		●
Swarm Linda		●				
Tagged sets	●					
T-Spaces		●		●		
TCP Linda		●		●		
TeenyLIME					●	
TIBCO ActiveSpaces		●				
The KLAIM family				●	●	
TinyLIME					●	
Triple Space Communication	●	●				
TuCSoN					●	
UML Spaces						●
VLOS	●					
Xcoordination Application Space and Xcoordination Coordination Space	●	●	●	●	●	
XMIDDLE					●	
XML Spaces				●		●
XVSM	●	●	●	●	●	●

Figure 2.11: Overview of SBC Implementations and Focus Groups

2.3 Space based computing implementations

In this section a selection of space based computing implementations is presented in detail in order to better understand the general concepts of space based computing implementations as well as their specialties. The main focus lies on the presentation of the general concepts.

2.3.1 Linda

Chapter 2.1.1 already introduced Linda[41] as the forbearer of all space based computing implementations. Hence, it makes sense to present Linda as the first implementation in this section to better understand the concept of space based computing.
David Gelernter and Nicolas Carriero explained in [CG90] that the main idea of Linda is based on the concept of generative communication *which attempts to unify the concepts of process creation and communication.*[42] Furthermore they presented a set of key facts which characterize Linda:

- Linda is not a programming language but a coordination language that defines a communication and synchronization model.
- In order to execute parallel processes Linda needs to be combined with a programming language, e.g. C, Java or Eiffel. The respective programming language provides *the semantics for computation while Linda provides the semantics for concurrency and computation.*[43]
- *Linda is fully distributed in space and distributed in time.*[44]

The Linda model uses the concept of tuple spaces. A tuple space is comprised of tuples. Linda uses a set of simple operations which allow the manipulation of the tuples in the tuple space [Kro00].
A tuple is a finite ordered sequence of typed fields where each field contains either a typed value or a process [EZCdrei92], i.e (“name“, 3, 1.5) is a tuple representing a sequence of the following values: A string “name“, an integer 3 and a floating point value 1.5. Such a tuple is called passive tuple because it only contains values.
Tuples which at least contain one process are called active tuples. An example for an active tuple is, i.e (“name“, 3, sin(x)) with sin(x) representing the process. Tuples which not only contain values (referred to as “actuals“) but also contain place-holders (referred to as “formals“) are called templates. *A formal is prefixed with a question mark.*[45] For example, (“name“, ?a, 3, ?b). The first and the third field are actuals and the second and the forth field are formals. Anti-tuples cannot contain processes because of restrictions imposed by the read and removal operations provided by Linda.
Linda provides 4 basic and 2 additional operations for accessing the tuple space:

[41] A trademark of the Associative Scientific Computing Associates
[42] [EZCdrei92] page 47.
[43] [EZCdrei92] page 47.
[44] [Gel85] page 1.
[45] [CG90] page 47.

- `out (t)`: adds a tuple `t` to the tuple space (non-blocking).
- `in (t)`: gets and removes a matching tuple `t` from the tuple space, blocks until a matching tuple is found.
- `rd (t)`: reads a matching tuple `t` from the tuple space, blocks until a matching tuple is found.
- `eval (t)`: places an active tuple in the tuple space, copied from the template. *When a process completes, it replaces itself within its respective tuple with the value resulting from its computation. When all processes within a tuple replace themselves with values, the formerly active tuple becomes passive.*[46]
- `rdp (t)`: Is like `rd (t)` but does not block, attempts to locate a matching tuple and returns 0 if it fails otherwise it returns 1 and reads the tuple.
- `inp (t)`: Is like `in (t)` but does not block, attempts to locate a matching tuple and returns 0 if it fails otherwise it returns 1 and removes the tuple.

Since tuples in Linda have neither a physical nor a virtual address the `in (t)` or `rd (t)` operations use associative matching to select a suitable tuple [EZCdrei92]. The next example shows how the operations `in (t)` and `rd (t)` work. We are looking for a tuple which matches with the following requirements:

- `in ('name', 3, ?a)`
- with `a` being of type integer

This means suitable tuples must contain the string "'name"' in the first field, the value 3 in the second field and any integer in the third field, e.g. ('name', 3, 7), ('name', 3, 367), ('name', 3, 1000). In case there is more than one matching tuple the final choice is made arbitrarily out of the set of suitable matches. The main difference between `in (t)` and `read (t)` is that `in (t)` removes the tuple from the tuple space and `rd (t)` leaves a copy of the tuple in the tuple space.
The next example illustrates what the operation `out ( 2, ?a)` does in detail, with `a` being of type integer again. In a first step the output `(2, ?a)` is placed into the tuple space and then can be taken by any `in (t)` operation *which has an actual of type integer in place of the formal*[47] `a`. A suitable example would be `(2, 4)` or `(?b, 7)`, if `b` is of type integer.
The `eval (t)` operation is very similar to the operation `out (t)` with the difference that `eval (t)` creates an active tuple in tuple space. This can be illustrated at the following example:
`eval (X (), Y ())` creates two processes `X ()` and `Y ()` *which are [...] evaluated concurrently.*[48] This means that the tuple is unavailable for matching as long as the two processes `X`

[46] [CG90] page 48.
[47] [EZCdrei92] page 49.
[48] [EZCdrei92] page 48.

`()` and `Y ()` are being evaluated. As soon as the evaluation of the two processes has finished, and a result is returned - for example two integer values 1 and 2 - the active tuple turns into the passive tuple (1, 2). eval creates new processes. This is how parallelism is created in Linda and from where the term generative communication is derived.
The two additional operations `inp (t)` and `rdp (t)` are the non-blocking versions of their counterparts `in (t)` and `rd (t)` [FP96]. If `inp (t)` and `rdp (t)` do not find a match in the tuple space they return the value false. If a match is found the value true is returned and then they execute like `in (t)` and `rd (t)` would do.
In his theoretical paper [Gel85] about Linda, Gelernter wrote that *the technique developed has attractive properties relative to other possibilities*[49] This is because Linda appears quite simple but at the same time it offers conceptually powerful possibilities to the programmer.
Although the Linda model was certainly not the solution to all the problems related to parallel programming, it was widely used and well accepted in the scientific community. Today Linda might not be state-of-the-art any more but its many successors illustrate the importance of the Linda model very well. During the classifaction process in chapter 4 we will also see where Linda has its weaknesses.

2.3.2 JavaSpaces

As explained in the previous section Linda was developed with the goal to be integrated in another host-language like C or Fortran. After its first appearance in the middle eighties, the concept of Linda fell more or less into disuse in the 1990s [Sch07]. However, with the development of the programming language Java[50], the growing importance of distributed systems and the need to communicate over networks, Linda gained in importance again. Hence it is no wonder that soon the idea arose to combine Java and Linda.
The result is called JavaSpaces[51] and it was developed by Sun Microsystems[52] based on the ideas of Linda. JavaSpaces is a part of the Jini[53], *intended to simplify the networking of heterogeneous systems.*[54]
In order to better understand JavaSpaces it is best to start with the explanation of the term entry. *A space stores entries.*[55] Basically an entry can be seen as the counterpart of a tuple in Linda. An entry is a typed group of object references represented by a class in the Java platform that implements the Entry interface as defined in the *Jini Entry Specification*[56]. Once an entry[57]

[49] [Gel85] page 108.
[50] http://www.java.com/en/
[51] http://river.apache.org/doc/specs/html/js-spec.html
[52] Sun Microsystems was bought by Oracle in 2010.
[53] Jini is a service orientated network architecture for the construction of distributed systems. It was introduced in 1998 by Sun and was transferred to Apache under the project name River in 2007.
[54] [WCC04] page 3.
[55] http://www2.sys-con.com/itsg/virtualcd/java/archives/0509/mahapatra/index.html
[56] http://river.apache.org/doc/specs/html/entry-spec.html
[57] The requirements each entry class must meet are described in detail in the Jini Entry specification under `http://river.apache.org/doc/specs/html/entry-spec.html` and are therefore not presented at this stage.

is created, different operations can be performed to interact with the space. Subsequently, the seven main operations will be presented in detail [Son03]:
The operations `write`, `read` and `take` can be seen as the basic operations in JavaSpaces and they can be compared to the operations `out (t)`, `rd (t)` and `in (t)` in Linda. The three operations are described below:

- `write`: Puts a copy of an entry into the space.
- `read`: Returns a copy of an entry matching a particular template(equivalent to anti-tuple). Read blocks in case a matching entry is not available.
- `take`: Returns and removes a matching entry from the space. Take blocks in case a matching entry is not available.

The operations `readIfExists` and `takeIfExists` are the non-blocking version of `read` and `take`. In fact `readIfExists` and `takeIfExists` work like the operations `read` and `take` with the difference that they return immediately returning a zero value *except in cases when a matching value exists within a transaction.*[58] Then they wait for the matching entry until it has finished the transaction or the timeout of the operations expires.
The operation `notify` *is used to register interest in the arrival of an entry in the space that matches a specified template.*[59] As soon as an entry has arrived that matches the template the caller gets notified by the space. In order to make this possible Jini introduces three entities: the event source, the remote event object and the remote event listener. In JavaSpaces the space acts as the event source that fires events *when entries are written into it and notifies processes that have registered interest in entries that match specified templates.*[60] This process is described in detail in the *Jini Distributed Event Specification*[61].
JavaSpaces also introduces the operation `snapshot` in order to reduce unnecessary overhead when the same entry is used over and over without modification [FAH99]. *For instance, it is common to use the same template multiple times in take or read operations, say within in a loop.*[62] In order to adress this issue the operation snapshot *returns a snapshotted version of the original entry, which can be used in other operations to avoid unnecessary serialization*[63]
Although this is an extremly handy operation, two things have to be borne in mind when using the operation `snapshot`:

- Any changes to the orignal template do not affect the snapshot.
- *A snapshot is only guaranteed to work with the space that created it.*[64]

[58] http://www2.sys-con.com/itsg/virtualcd/java/archives/0509/mahapatra/index.html
[59] http://www2.sys-con.com/itsg/virtualcd/java/archives/0509/mahapatra/index.html
[60] http://www2.sys-con.com/itsg/virtualcd/java/archives/0509/mahapatra/index.html
[61] http://river.apache.org/doc/specs/html/event-spec.html
[62] [FAH99] page 48.
[63] [FAH99] page 48.
[64] [FAH99] page 49.

JavaSpaces not only provides more operations than Linda it also extends the Linda model in order to address some of the existing limitations of Linda. The main enhancements can be seen in the introduction of new operations (notify and snapshot), the concept of leasing and transactions via Jini. A downside one could argue is that JavaSpaces does not have an `eval` operation like Linda.

Leasing is a concept that allows resources to reserve entries in the space for a specified period of time [Sch07]. All resources in Jini have a lease. Every entry in the space is tied to a lease which means after its lease has expired or has been revoked the entry is deleted from the space. In a distributed environment, the use of leases is beneficial as unnecessary or orphaned entries are deleted after a certain time and do not block unneeded resources. When writing an entry in the space a parameter which states the requested lease time is assigned to the entry in order to specify how long the entry shall be stored in the space, e.g. write (anEntry, null, 1000). In this example the entry shall be stored for 1000 milliseconds in the space before it gets deleted. *The write operation actually returns a lease object representing the lease time granted by the space, which may be less than the requested lease time.*[65] In case that the granted lease time is too short, the lease time can be extended before its expiry. The responsibility for renewing the lease time of an entry lies with the process who owns the entry. Again the space decides for how much time longer an entry can stay in the space which can lead to several renewal operations in order to obtain the desired lease time. In order to make sure that an entry stays in the space for an infinite lease time one can use the constant Lease.FOREVER. Lease times can also be applied to transactions. The concept of leasing is described in detail in the *Jini Distributed Leasing Specification*[66].

Transactions play an essential role in JavaSpaces which provides a distributed transaction service through the Jini transaction model which is described in detail in the *Jini Distributed Transaction Specification*[67]. Chapter 4.8.3 will describe in more detail why transactions are an essential tool for all distributed systems *which need to operate safely and correctly in the presence of partial failure.*[68] The main idea of transactions is that a set of single operations can be grouped together in a way that they can be executed together. Once being grouped such a transaction is only *performed atomically which means either all operations complete or none of them.* [69] In case a process wants to carry out a transaction it *asks a transaction manager to create a transaction and manage it for a specified lease time.*[70] As soon as the process has obtained the transaction from the transaction manager it passes the transaction to each operation that should be part of it. After having successfully grouped all operations the process commits or aborts the transactions which results in the completion of all operations.

JavaSpaces has a significant number of implementations including amongst others Blitz[71] and GigaSpaces[72]. A detailed comparison of different implementations within the Java family can

[65] http://www2.sys-con.com/itsg/virtualcd/java/archives/0509/mahapatra/index.html
[66] http://river.apache.org/doc/specs/html/lease-spec.html
[67] http://river.apache.org/doc/specs/html/txn-spec.html.
[68] http://www.javaworld.com/jw-04-2000/jw-0421-jiniology.html?page=6
[69] http://www.javaworld.com/jw-04-2000/jw-0421-jiniology.html
[70] http://www.javaworld.com/jw-04-2000/jw-0421-jiniology.htm
[71] `http://www.dancres.org/blitz`
[72] `http://www.gigaspaces.com/`

be found in chapter 4.2.2.

2.3.3 LIME

LIME[73] [PMR99] is based on the Linda model and implemented in Java. We have already learned that Linda supports decoupling in space, time and reference which makes Linda also interesting for highly dynamic environments. Such environments have to deal with dynamic communities where *changes in connectivity among hosts are a normal occurrence as components move in and out of range of one another*[74]. The content in such a community is composed of the data, available at a certain time, of all individual members of the community [PTW02]. LIME is capable to adress both the challenges of logical and physical mobility.
Before we start to look into the details of LIME it is important to explain a few terms and key facts:

- *Lime agents are the only active entities in Lime and they can roam across mobile hosts. Every agent owns its own Lime tuple space which is permanently connected to the agent and refered to as interface tuple space (ITS).*
- *Mobile hosts serve as containers for the agents and can roam across a physical space.*
- *Lime has not a persistent tuple space.*[75]
- In order to support a scenario based on mobility, LIME divides the tuple space into different tuple spaces which are associated permanently with mobile agents [Col01]. These single tuple spaces form when they are merged, a federated transiently shared tuple space[76], which creates the illusion of one single tuple space *containing the tuples of all agents on the different hosts*[77].

Figure 2.12 summarizes the key facts given before and brings them into context. Agents run on mobile hosts where each agent has access to its personal *interface tuple space that is permanently associated with that agent and transferred along with it when movement occurs.*[78]. Of course agents can have more than one tuple space (public and private ones e.g.). The interface tuple space contains the tuples which the agent wants to share with other agents and allows each agent to carry out operations which are identical to those presented in Linda with the difference that LIME does not have an `eval` operation.

[73] Linda in a Mobile Environment
[74] `http://lime.sourceforge.net/Lime/nutshell.html`
[75] `http://lime.sourceforge.net/Lime/nutshell.html`
[76] Transient, because the content of the tuple space varies depending on the migration of the agents in and out of the tuple space
[77] `http://lime.sourceforge.net/Lime/nutshell.html`
[78] [PM99] page 370.

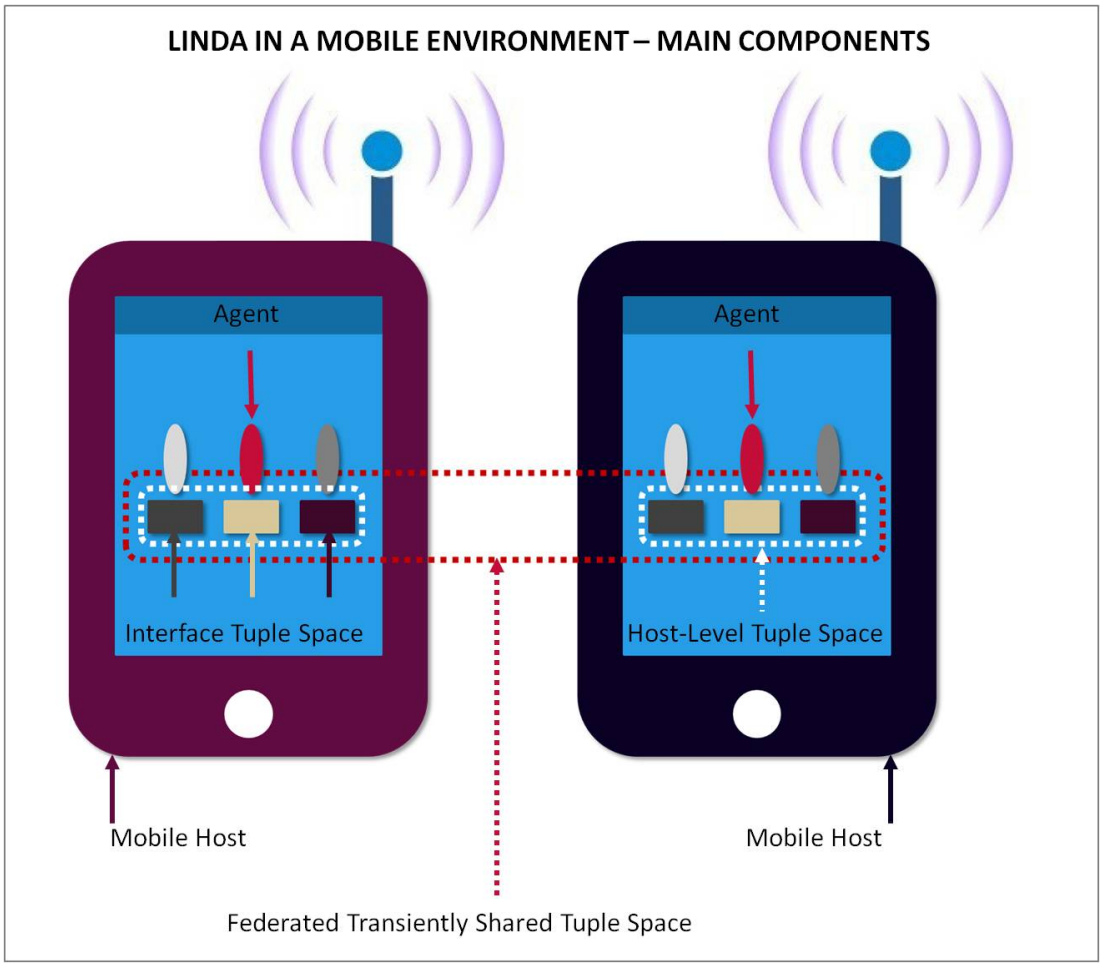


Figure 2.12: Linda in a Mobile Environment - Main Components

A published tuple space has depending on its status access to the corresponding host-level tuple space or federated transiently shared tuple space [Pri02]. The agent is aware of the scope of its environment, e.g. the number of hosts and agents, and can carry out its operations transparently. The concept of transiently shared atomic tuple spaces entails that the content of a tuple spaces from the perspective of an agent changes dynamically.
The processes of growing and shrinking the federated transiently shared tuple space are referred to as engagement and disengagement [PMR00]. Both processes occur as transactions. Figure 2.13 illustrates this processes with the help of a simple example in three steps. Firstly four mobile hosts within reach establish connectivity and engage with each other. All four mobile hosts have now access to the public spaces of the other mobile hosts which is shown in step two. One mobile host (bottom right) leaves the federated transiently shared tuple space through disengagement. After the disengagement this mobile host has only access to its own tuple space as shown in step three.

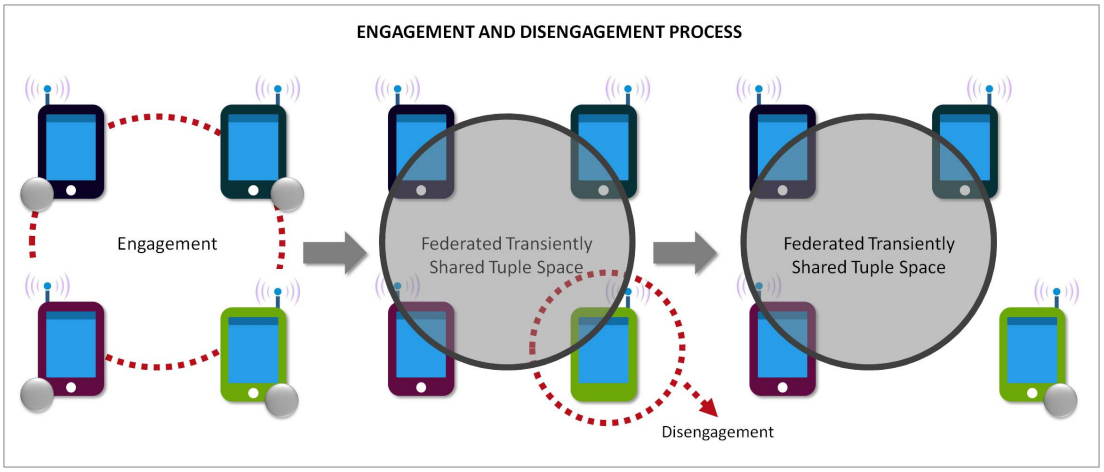


Figure 2.13: Engagement and Disengagement Process

Besides the standard Linda operations LIME introduces two other concepts which will be discussed separately hereinafter:

- tuple locations
- reactions

Tuple locations. An agent can be identified through its so called location which is composed of the network address of the host and a global unique ID for the host [Pri02]. Without adding a location parameter to an operation, LIME will take all tuples of all agents in the federated transiently shared tuple space into account for the respective operation. This fact allows LIME to be not dependent on the knowledge of the location of its agents which can be seen as an advantage. However, adding such a location parameter the scope of tuple space operations will be restricted to the location [CVV04].
The following example[79], accompanied by figure 2.14, illustrates the mode of operation of the location parameter. To write a tuple in a particular tuple space of another agent the operation `out[ λ] (t)` can be used, which extends the basic `out (t)` operation to the parameter λ. The parameter λ indicates the target location. The operation `out[ λ] (t)` is executed in a two step process:

- In a first step the tuple `t` will be placed in the private tuple space of the executing agent ω. The tuple `t` carries now the information of two locations: Its current location ω and its destination location λ.
- If the agent λ is available, the tuple `t` is moved from ω to λ. This process can be executed also with a delay if agent λ is not available immediately.

[79] [Pri02] page 5.

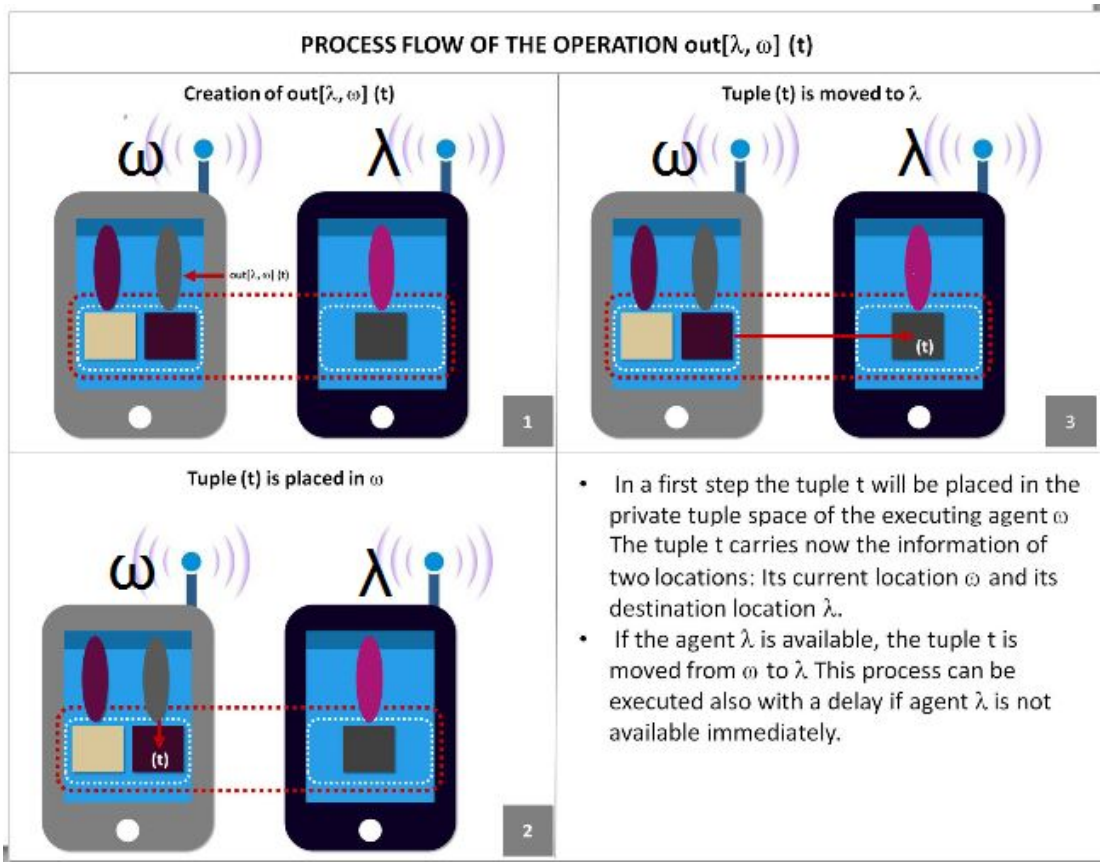


Figure 2.14: Process Flow of The Operation out[λ, ω] (t)

Apart from the `out[`λ`] (t)` operation LIME also allows to add location parameters to the operations `in[`λ`] (t)`, `inp[`λ`] (t)`, `rd[`λ`] (t)` and `rdp[`λ`] (t)`.

Reactive programming. LIME is able to respond to certain events, such as the arrival or departure of a mobile agent and the connection or disconnection of a mobile host e.g. [Col01]. The operations `in (t)` and `rd (t)` already allow responding to some of these events but as both operations block a thread per event this is neither a cheap nor efficient solution. In order to address this problem LIME introduces reactions. A reaction `R (`*`s`*`, `*`p`*`)` is bound to a code fragment `s`, which is executed in case that template `p` is found in the corresponding tuple space [PTW02]. Each time an operation is executed on a tuple space all the associated reactions for this tuple space are checked for matching templates `p` in no particular order. In case a suitable `p` is found, the code fragment `s` is executed. In case a reaction refers to an entire federated transiently shared tuple space, resources to carry out a reaction are normally very high (depending on the size of the federated tuple space). In order to enhance performance LIME offers two types of reactions:

- Weak reactions: Operations are not performed atomically. Atomicity in this context describes the ability to perform multiple operations on a diverse set of hosts in a single

atomic step.

- Strong reactions: *The operation is performed atomically, i.e., even if some of the reactions become suddenly enabled by the current state of the tuple space, none can fire until all of them have been registered. The reactions must have current and destination location fields.*[80]

Chapter 4.3.4 will discuss the strengths and weaknesses of LIME in context to other solutions in more detail.

2.3.4 Triple Space Communication

This chapter presents Triple Space Communication which plays a special role amongst space based computing implementations. It can be seen as a visionary approach in the field of space based computing implementations because it combines several technologies in order to create a *network that connects applications based on machine-processable semantics of data.*[81]. In other words Triple Space Communication aims to *become the web for machines as the web based on HTML became the Web for humans.*[82]
Triple Space Communication was a strategic research project which was funded by the European Commission between 2006 and 2009. The project was supported by partners from academia[83] and industry[84].
Before we look in more detail at Triple Space Communication itself it makes sense to explain its general concept. Triple Space Communication combines three basic technologies:

- tuple space computing: In the case of the Triple Space Communication the tuple space technology provides a virtual shared memory which ensures the persistent communication process. The data is presented in atomic units called triplets[85].

- semantic web: The semantic web component helps to add *machine-processable semantics to data. The computer can "understand" the information and therefore process it on behalf of the human user.*[86]

- web services: Web services[87] in Triple Space Communication really follow the web paradigm of 'persistently publish and read' [Bus05] which means that data can be published at any time, at any location and without knowing who is going to read the data in

[80] `http://lime.sourceforge.net/Lime/api/lime/LimeTupleSpace.html`
[81] `ftp://ftp.cordis.europa.eu/pub/ist/docs/kct/tripcom-pr-poster-jun06_en.pdf`
[82] `http://www.tripcom.org/description.php`
[83] Leopold Franzens University Innsbruck, National University of Ireland, Galway, University of Stuttgart, Vienna University of Technology and the Free University of Berlin.
[84] Ontotext Lab, Sirma Group Corp., Profium OY, CEFRIEL SCRL. and Telefonica I+D.
[85] Normally tuples.
[86] [Fen04] page 43.
[87] Normally web services follow a synchronous communication where the sender and receiver have to know each other in order to exchange data.

> the future. In the case of Triple Space Communication, web services help to display the *heterogenous components into an unified way.*[88]

So Triple Space Communication is a way to help machines to communicate with each other. Once this topic is completely explored Triple Space Communication will be a fast, efficient and secure service adressing a big variety of use cases. A very interesting use case for Triple Space Computing could be in the sector of eHealth for the European Patient Summaries which is shown in [CVF+07].
After having looked at the general vision and the concept of Triple Space Communication, the next part will focus on more technical aspects.
Triple spaces share a lot of characteristics with tuple spaces. At the beginning triple spaces are empty. The data in the triple space is presented through RDF[89] triples. RDF triples can be explained as tuples with three fields, called triples, which have the following form `<subject, predicate, object>` [NCV+07]. Each field of the triple contains an URI[90]. The reason for chosing the RDF triples over the standard tuples is that *graphs are used as the main data structure for communication*[91] which offers a more expressive data model. The following operations are offered by Triple Space Communication [NCV+07], [TN08]:

- `void out(Triple t, URI space)`: A triple is written into the addressed triple space.
- `set<Triple> rda(Template t, URI space, integer timeout)`: One triple which matches the template `t` and lies within the specified space is returned. The timeout obviates too long blocking periods.
- `set<Triple> rda(Template t, integer timeout)`: One triple which matches the template `t` is returned. The template `t` can lie in any triple space. Therefore a lot of different answers are possible.
- `set<Triple> rd(Template t, URI space, integer timeout)`: An arbitrary number of triples which match the template `t` are returned. The other specifications are the same as for the operation `rda`.
- `set<Triple> rdg(Template t, URI space, integer timeout)`: *Returns the entire content of a named graph that contains a matching triple.*[92]
- `set<Triple> ina(Template t, URI space, integer timeout)`: Like `rda` with the difference that it destroys the matching triple.
- `set<Triple> in(Template t, URI space, integer timeout)`: Like `rd` with the difference that it destroys the returned triples.

[88] `ftp://ftp.cordis.europa.eu/pub/ist/docs/kct/tripcom-pr-poster-jun06_en.pdf`
[89] Resource Description Framework
[90] Universal Resource Identifiers
[91] [NCV+07] page 2.
[92] [NCV+07] page 3.

- `set<Triple> ing(Template t, URI space, integer timeout)`: Like `rdg` with the difference that it destroys the entire content of the returned graph.
- `set<Triple> subscribe(Template t, URI space)`: A process can subscribe to a particular type of information by providing a template and will be notified as soon as a matching triple is put into the specified space.
- `set<Triple> unsubscribe(Template t, URI space)`: This operation cancels a subscription.
- `boolean: create(URI space, [URI parent, URI transaction])`: This operation creates a new space, as child of its parent space.
- `boolean: destroy(URI space, [URI transaction])`: This operation destroys the specified space with all its *subspaces and all contained triples.*[93]

To bring the idea of Triple Space Communication into life, three main components need to be introduced, that operate together in a Triple Space [Bus05]:

- The first system which will be introduced are Triple Space Clients. Clients can read and write triples.
- Clients communicate with the different triple spaces through the so called *Triple Space Transfer Protocol*[94].
- Triple Spaces lie on a triple space server that can host any number of triple spaces.

Looking at the big picture the Internet allows an arbitrary number of triple space servers whereas each of them can host *hundreds of triple spaces.*[95] Figure 2.15[96] illustrates the system elements and boundaries.

[93] [NCV+07] page 3.
[94] TSTP
[95] [Bus05] page 6.
[96] [Bus05] page 6.

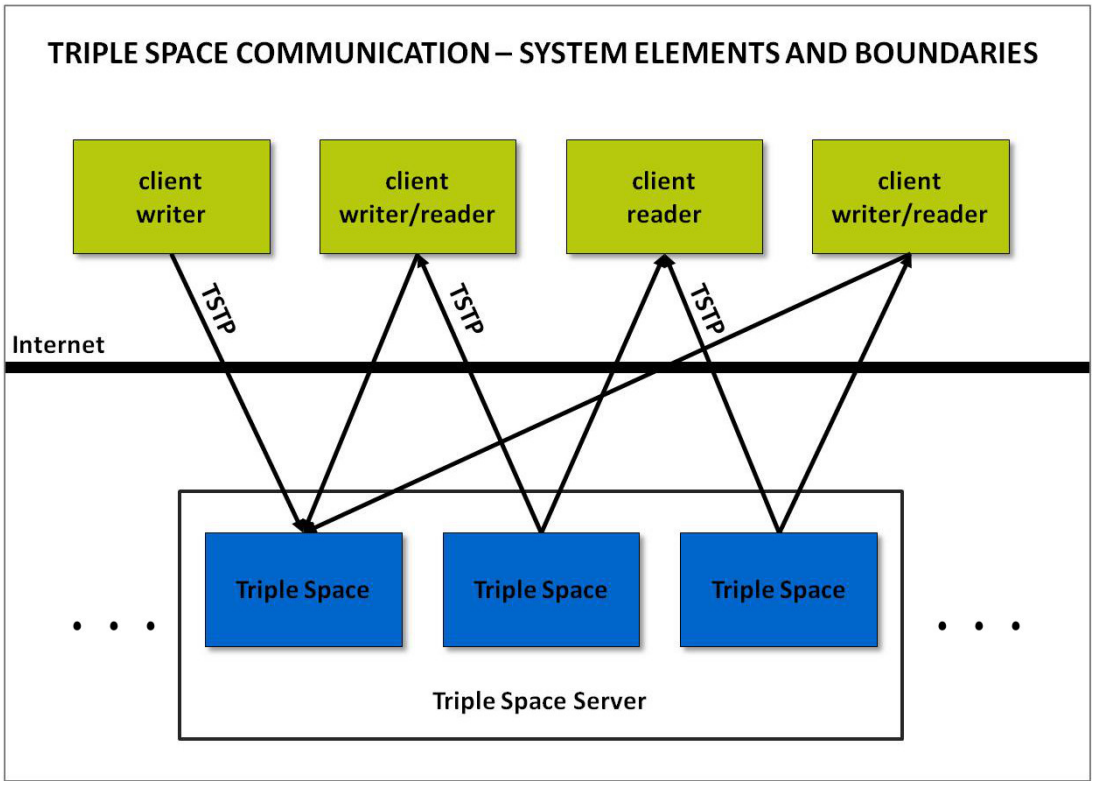


Figure 2.15: Triple Space Communication - System Elements and Boundaries

The triple space architecture as outlined above is a very basic one which aims to illustrate how asynchronous communication for machine-to-machine communication can look like. The important aspect of triple space communication is that is able to be autonomous in *time, space, reference, and data schema.*[97] The big advantage though is that writers can persistently publish data and readers can read the available data without knowing the writer.

2.3.5 TuCSoN

The concept of TuCSon[98] is quite similar to the concept of LIME [Sau08], which has already been presented in section 2.3.3. TuCSoN allows to distribute tuples and share them between different tuple spaces. TuCSoN, as defined on its website, *is a Java-based*[99] *model for the coordination of distributed processes, as well as autonomous, intelligent and mobile agents.*[100] The major innovation is the introduction of the ReSpecT[101] language, which allows to define the behaviour of tuple centres in response to different communication events [Omi06]. But

[97] [Bus05] page 12.
[98] Tuple Centres Spread over Networks
[99] also Prolog-based
[100] `http://apice.unibo.it/xwiki/bin/view/TuCSoN/`
[101] Reaction Specifcation Tuples

before these specifics are discussed in detail, a general overview of of the basic infrastructure and features of TuCSoN is given.
I will start with a definition of the most relevant entities which are relevant to understand the ideas and functionalities of TuCSoN:

- **Tuple centres.** Instead of tuple spaces, as introduced by Linda, TuCSoN uses so called tuple centres as coordination medium [COZ00]. A tuple centre has exactly the same characteristics as a tuple space, with the only difference that a tuple centre offers *the possiblity to program its behaviour in response to interactions events.*[102]

- **Agents.** *Agents are the coordinatables*[103] which means they take part in the coordination process. In other words basic TuCSoN agents have the ability to access the TuCSoN infrastructure. Besides the basic TuCSoN agents other agents like IDE agents, GUI agents and the Inspector agents exist [ROD01]. They are used for deployment, debugging and monitoring purposes.

- **Nodes.** *Each tuple centre is associated to a node*[104], which corresponds to Internet hosts or servers connected over the network [Omi06]. A node can contain an arbitrary number of tuple centres. One can distinguish between two types of nodes [NOVS11]:

  - **Gateways.** A gateway node offers agents the possibility to access *other nodes and their tuple centres.*[105] The role of gateways is fundamental to support and control the execution of the agents moving along the network or remotely accessing places. *Each gateway can authenticate agents on behalf of its associated domain, by verifying agents' identity and by propagating it by default to all domain's places and subgateways. More precisely, agent authentication must be performed by each gateway receiving an agent request from an external source, as usual in Internet applications with security requirements.*[106]
  - **Places.** A place node hosts tuple centres for the specific applications / systems, e.g coordination activities.

- Domains. A domain is composed of gateway and place nodes.

Agents as well as tuple centres are spread over the network [OM12]. The agents can move across the network whereas tuple centres are bound[107] to the device that they run on. Now one can connect the different components and see that TuCSoN creates a system where *(possibly mobile) agents and (possibly mobile) tuple centres are coordinating in a (possibly) distributed set of nodes.*[108] Figure 2.16 visualizes this system view.

[102] [Omi06] page 5.
[103] [OM12] slide 6.
[104] [AD99] page 3.
[105] [COZ00] page 5.
[106] [COZ00] page 80.
[107] In case the device is a mobile one the tuple centres inherit this mobile quality.
[108] [OM12] slide 8.

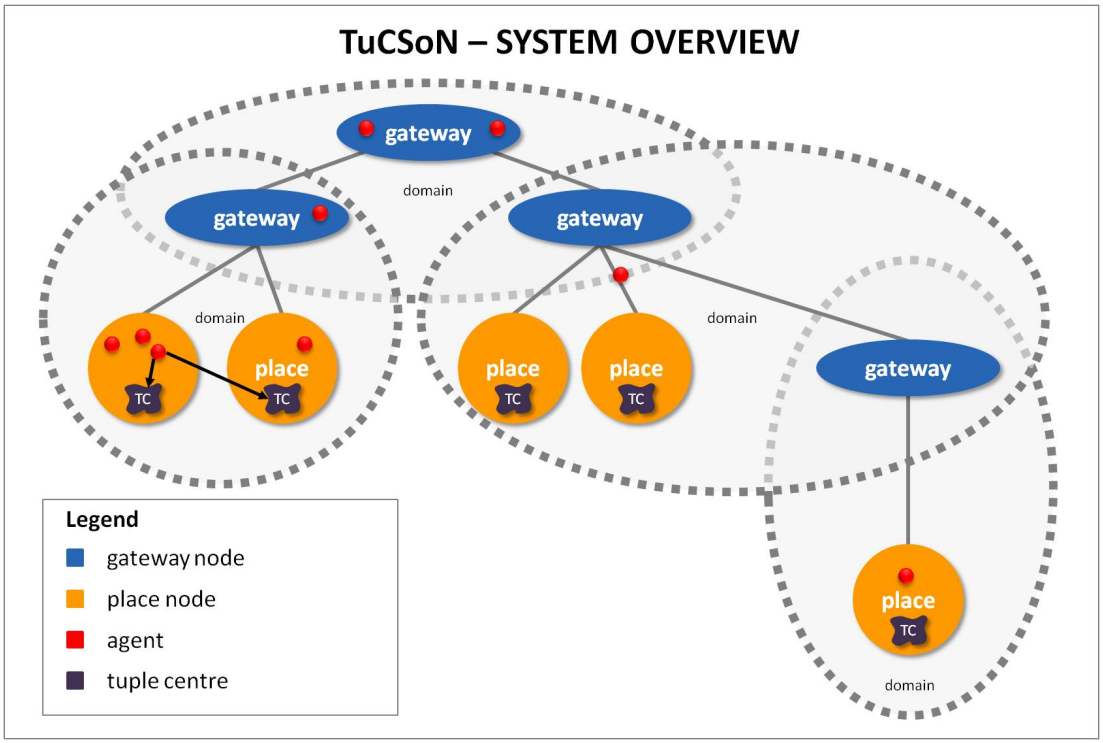


Figure 2.16: TuCSoN - System Overview

Another aspect which figure 2.16 shows is that by connecting the gateway nodes in a tree-like way, it makes it easy to provide different security characteristics for different domains. Another key-characteristc of TuCSon is that all agents, nodes and tuple centres have unique identities which helps to adress them easily [OM13]:

- Nodes are identified through the following two parameters - their network ID[109] and their port number. <NetworkID, PortNo>.
- Every tuple centre has a unique name that identifies it. In order to adress/identify a tuple centre one has to know the exact node <NetworkID, PortNo> the tuple centre is on and the name of the tuple centre itself.
- Agents are identified through their name and a UUID[110] which is assigend to the agent when it first enters a TuCSon system.

TuCSoN, as Linda or LIME, offers a set of primitives which agents can use in order to interact with the different tuple centres. A few basic operations are already known from Linda and therefore not discussed in this section. Additional operations are described and illustrated for better understanding.
New operations in TuCSoN are described below [NOVS11]:

[109] Either IP adress or DNS entry of the hosting device
[110] Universally Unique Identifier

- `no (TupleTemplate)` looks for a tuple `t` matching tuple template in the target tuple space and if no matching tuple is found when the operation is served, the execution succeeds, and the tuple template is returned. Otherwise, the execution is suspended to be resumed and successfully completed when no matching tuples can any longer be found in the target tuple space, then the tuple template is returned.

- `nop (TupleTemplate)` is the predicative version of `no (TupleTemplate)`. If a matching tuple `t` is found the execution fails and `t` is returned.

- `get ( )` reads all the tuples in the specified tuple centre and returns them as a list. If no tuple occurs in the specified tuple centre at execution time, the empty list is returned and the execution succeeds anyway.

- `set (Tuples)` overwrites the specified tuple center with tuples on the list (which has been derived from a `get ()` operation). When the execution is completed, the list of tuples is successfully returned.

After having having discussed the basic model and operations it is time to focus on a key feature which distinguish TuCSoN from Linda, namely its programmability [NOVS11].
Programmability. It has been already pointed out that one key strength of TuCSoN is that tuple centres are programmable, in other words they can *react to incoming/outgoing communication events*[111]. Programmability is achieved through so called reactions which can access and modify the involved tuple centre as well as its tuples during an event [OZ98b]. An important fact is that a reaction is a sequence of logic operations which either succeeds if all its reaction operations succeed, and fails otherwise. Reactions are *executed sequentially with a transactional semantics: so, a failed reaction has no effect on the state of a tuple centre.*[112]
In the case of TuCSoN this transaction is handled with the first-order logic language ReSpecT that is responsible for making tuple centres programmable. ReSpecT allows events to be associated to certain reactions through the introduction of so called specification tuples, of the form reaction (E,G,R). E stands for a communication event that is associated with a reaction R in case all conditions G are satisfied [NOVS11].
Since TuCSoN is one of the long-living space based computing implementations[113] in the market it is pretty clear that it has been improved and enhanced over the years. TuCSoN adresses amongst others topics like access control [COZ00] and semantic support [NVP10], [NCOA10]. Later chapters especially 4.10.1 and 4.12 will adress these topics in more detail.

2.3.6 XVSM

XVSM[114] [KRJ05] is a space-based middleware developed by the Space Based Computing Group of the Institute of Computer Languages at the Vienna University of Technology. It combines the idea of tuple spaces with additional, flexible functionalities, called aspects, in order

[111] [NOVS11] page 4.
[112] `http://alice.unibo.it/xwiki/bin/view/ReSpecT/Overview`
[113] TuCSoN was introduced in the late 90s.
[114] eXtensible Virtual Shared Memory

to provide exactly the functionality the user needs. XVSM allows processes to collaborate with each other in an easy way [KC12]. Another speciality of XVSM is that these processes can exchange information through different coordination mechanisms [Win11]. Later in this chapter, aspects as well as the different coordination mechanisms will be discussed in more detail, but for now the very basics of XVSM are presented.
An XVSM space is typically composed of the following components [Bar10]:

- The main component of the XVSM space is the so called XVSM Core. Every machine in the space has its own core[115]. A core manages and contains so called containers.
- Containers hold entries, which are managed by one or more coordinators. One can distinguish between local and remote containers. Each container can be adressed or accessed by an unique URL.
- Coordinators offer different types of coordination patterns, e.g. Linda, FIFO[116], Index, Key et cetera.
- *Entries are the representation of user-objects in the space*[117] and can be compared to tuples in Linda.

Figure 2.17 gives an example of how such a XVSM space could look like. It also shows how users can access remote respectively local containers.

[115] Note that also several cores on one machine are possible.
[116] First In First Out
[117] [Bar10] page 6.

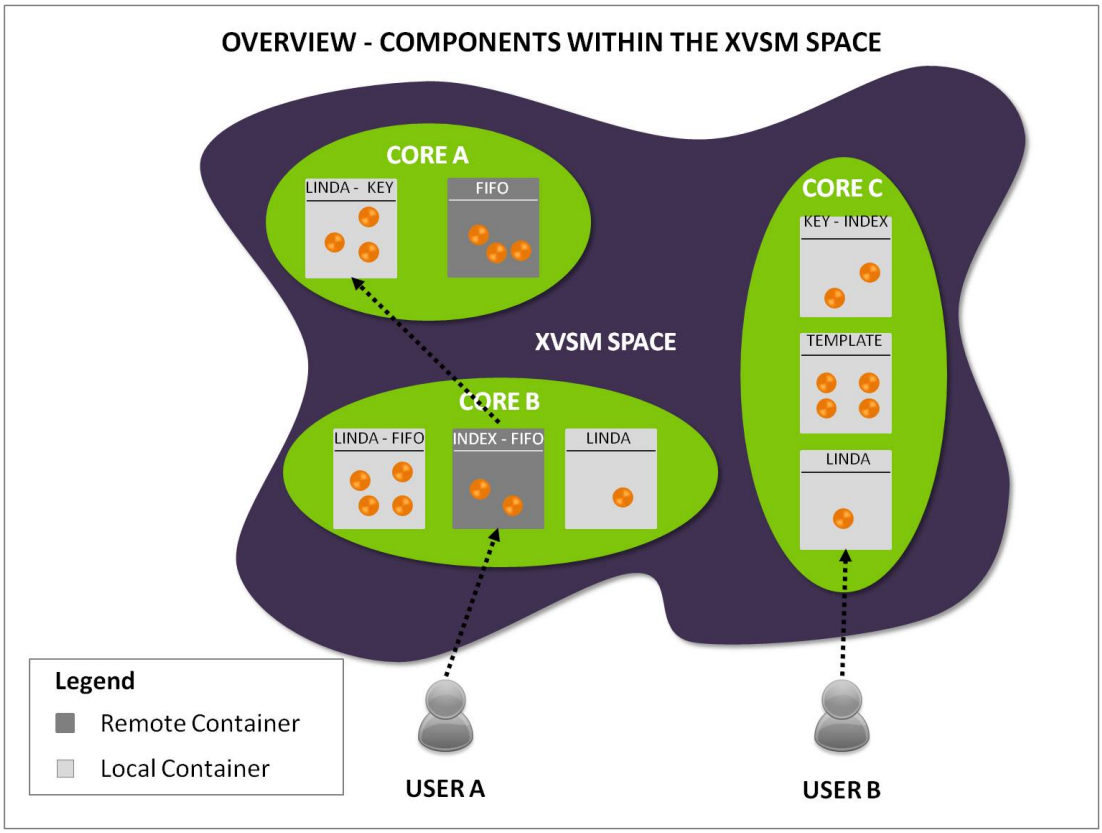


Figure 2.17: Overview - Components within the XVSM Space

In order to make interaction as shown in figure 2.17 possible, XVSM provides a set of basic operations on containers (create, publish and destroy) and on entries (read, take, write and destroy). XVSM offers two other interesting features in context with operations. First of all containers in XVSM *support bulk operations, so that it is possible to insert multiple entries into a container respectively to retrieve/remove multiple entries out of it within one operation.*[118] Secondly, XVSM operations are not responsible for blocking mechanisms. Whether an operation blocks depends only on the used coordinator [Mor10].
One of the main differentiators of XVSM over other space based computing implementations is the focus on the extensibility of the coordination [Pro08]. The entries in the containers are organized by coordinators, which are attached the respective containers. As a matter of fact the coordinators in the containers are able to deal with different coordination types and can manage a lot of tasks, e.g. *provide explicit and direct access to entries via keys or labels.*[119] The XVSM system can be extended by user defined coordinators can be always extended through different coordination mechanisms which offers good flexibility.

[118] [Mor10] page 72.
[119] [Pro08] page 16.

XVSM provides amongst others the following built-in coordinators[120] [121] which are described consecutively [Pro08]. Chapter 4.4 will describe these coordination concepts in more detail:

- **Key Coordinator**: The coordination type key allows to define a key for every entry. Typically the keys are strings. In order adress a specific entry you have to use the matching key.
- **FIFO Coordinator**: The FIFO coordination can be compared with a queue. FIFO stands for a first-in, first-out coordination mechanism which means that the entries are retrieved in the same order as they were written to the container.
- **LIFO Coordinator**: LIFO stands for last-in, first-out and shows exactly the opposite behaviour than FIFO. The entry which was last written to the space is taken out first.
- **Linda Coordinator**: The Linda Coordinator uses the Linda template matching mechanism to retrieve entries out of a container. A template is compared against all the entries (tuples) within the container and suitable matches are returned.
- **Label Coordinator**: *With the Label-Coordinator, entries are accessible by labels. This approach is quite similar to the Key-Coordinator, however, labels do not have to be unique, in contrast to keys. This allows multiple entries to have the same label.* [Bar10]

Aspects. The behaviour of a container can be extended through aspects[122] which react on certain events when an operation is carried out on the container [KMG+09]. Aspects as shown in figure 2.18[123] extend the core in a flexible way and address topics like encryption, authentication, monitoring, lifecycle management, messaging, replication or persistency.

[120] `http://www.complang.tuwien.ac.at/xvsm/1.0-alpha/docs/api/org/xvsm/interfaces/ICoordinator.html`

[121] `http://www.complang.tuwien.ac.at/xvsm/1.0-alpha/docs/tutorial/MozartSpaces_Tutorial.pdf`

[122] Aspects are basically code fragements.

[123] `http://www.complang.tuwien.ac.at/eva/SBC-Group/sbcGroupIndex.html`

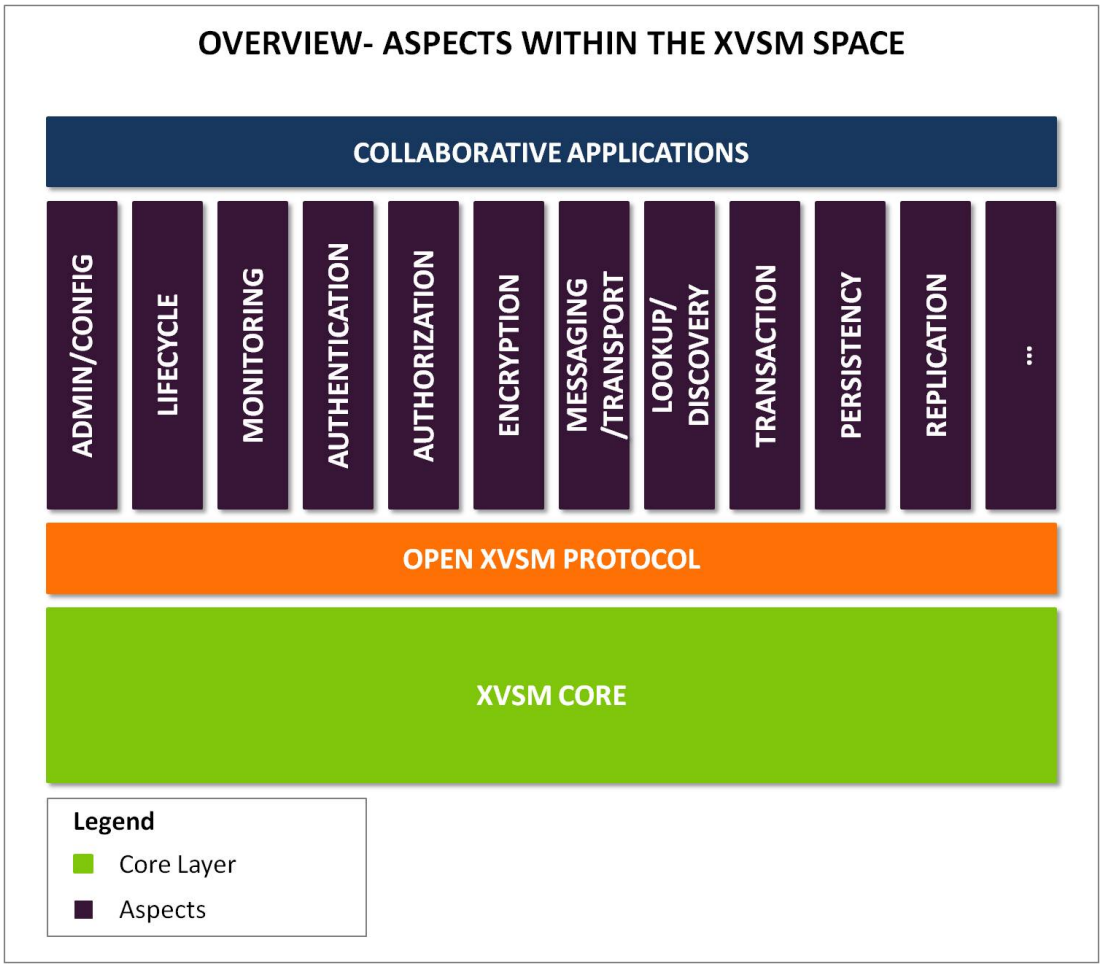


Figure 2.18: Overview - Aspects within The XVSM Space - Image taken from the SBC-Course

XVSM has been implemented in different programming languages [Mor10]. On the one hand it has been implemented using Java and is called MozartSpaces[124] [Pro08], [Sch08c]. The .Net implementation is called XCoSpaces[125] ([Sch08b], [Kar09]). A new descendent of XVSM is LinqSpace [Gel11] which extends the XVSM middleware solution with the .NET environment LINQ[126] with the aim to enrich XVSM *with uniform and versatile query capabilities.*[127] Chapter will show that XVSM is an implementation which is continuously improved, e.g. [KRML08b] and information and publications on it can be found on the project website[128] which offers also publications.

[124] `http://www.mozartspaces.org`
[125] `http://www.xcoordination.org`
[126] Language Integrated Query
[127] [Gel11] page 3.
[128] `http://www.xvsm.org`

CHAPTER 3

Application scenarios

Chapter 3 presents possible application scenarios for space based computing implementations. Although the application scenarios have been divided into different categories it is important to understand that the application scenarios presented might overlap. The presented application scenarios have been choosen because the used literature has been focusing on these application scenarios as well.

3.1 Scenarios focusing on near-time-data distribution

Having the right data at the right time is important for a lot of applications, e.g. data warehouses, stock trading platforms [KBM05] or communication via satellite. The latter one is especially challenging [WG06] since we expect working broadband communication even if we are in moving trains, airplanes or in a jeep in the Sahara. Fullfilling this expectation is hard since we move, want to connect to an possibly unreliable network and the distance between us and the next satellite is not within a stone's throw. For the sake of simplicity we will focus on a more down to earth application scenario in this chapter.
In the case of data warehouses, enterprises want to be able to respond to near-time business events as they occur [TKA10]. But still many enterprises struggle to get the right information fast enough and even worse the information is often inconsistent. This problem is mainly caused by the fact that in large industry companies or organizations there is no such thing as a single data-warehouse which represents the single-point of truth. Instead it is more likely that one is confronted with various databases of all sorts, producers and ages. And as if this would not be worse enough the different databases are most likely arbitrarily distributed over the heterogenous entities of the enterprise [Kue03].
Imagine you are a car manufacturer and you have discovered that one of your production lines is working slower than it should do. It is only normal that you want to check why this can happen and you might want to look at relevant numbers to get to the roots of the problem. Now you have to rely on your data warehouse which needs to give you appropriate information

without long data latencies. Also you want to be sure that the information provided is consistent because otherwise you may not find the cause of the problem or are most likely to make a wrong decision. Consequently near-time data distribution *has become a critical component of strategic and operational decision-making during the last years.*[1]

[Kue03] presents a business scenario for an Austrian bank which wants to synchronize data between subsidiaries based in different countries and in compliance with Basel 2. To realize this scenario [Kue03] introduces GONG[2][3] *which is based on the CORSO middleware.*[4] In this example the main data warehouse is located in Austria and the subsidiaries in Italy, Germany and Hungary. The main advantages of GONG are that there are no additional hardware costs and even more importantly that it supports the publish/subscribe pattern which allows every subsidiary to define which data they want to publish and/or subscribe. Furthermore subsidiaries can decide how often they want to synchronize the respective data, e.g. in real-time, as batch or whenever the data is needed. Figure 3.1[5] illustrates the concept.

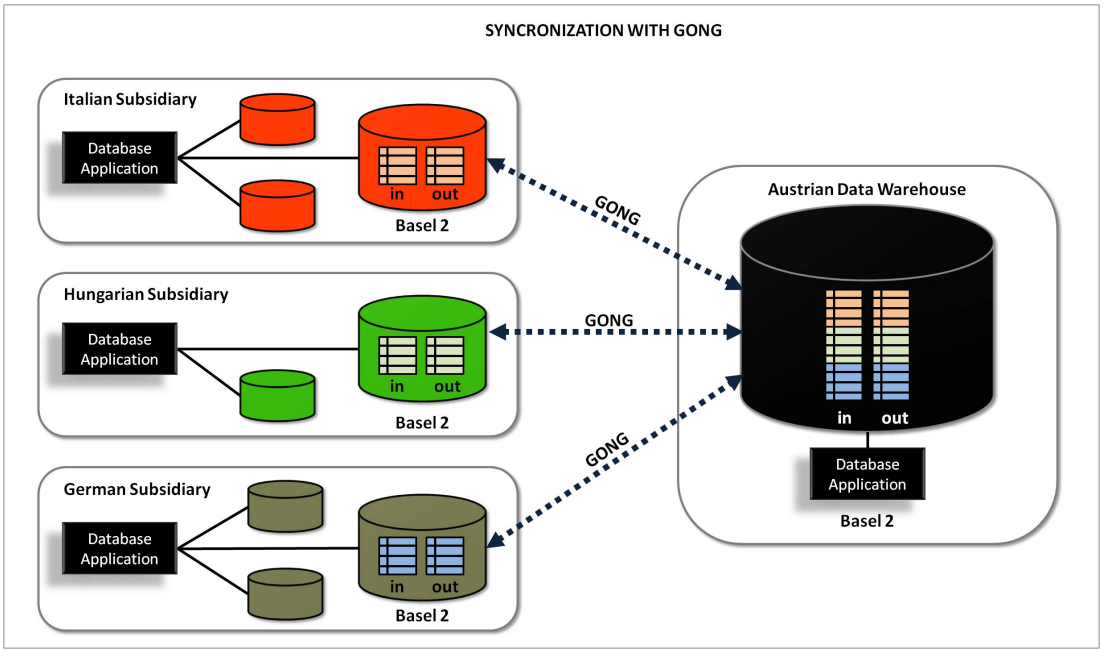


Figure 3.1: Synchronization with GONG

To finalize this chapter, it is important to understand that near-time data distribution or real-time data distribution [LPKL04] is although still sometimes a challenge alaready a standard we are used to. Therefore near-time data distribution in space based computing implementations is

[1] `http://www.martinsights.com/?p=147`
[2] General Database Notification Gateway
[3] GONG is a product by tecco Software Entwicklung AG.
[4] [Kue03] page 3.
[5] [Kue03] page 5.

often interlinked with other application scenarios which focus on other topics but still demand near-time data distribution.

3.2 Scenarios focusing on database replication

Every enterprise wants to be in full control over its data. However, within the last years an unprecedented growth in data could be denoted which subsequently led enterprises to the point where they lost full control of it [Daw12]. But why is control over data so important? Having control over data allows enterprises to better understand their business and make reasonable decisions based on the available data.
Apart from having control over their data, enterprises have quite a number of additional requirements when it comes to their systems. For example, scalability and availability are always of great interest for enterprises because both topics stand in direct concjunction with costs and the quality of service. In the area of mobility enterprises might also want to exchange data with mobile users. In some business fields e-commerce solutions are a must-have and represent a strong interface to the world of the customer. But how can an enterprise make sure that it systems scale well, are available at any given time and are fault tolerant? In this context database replication, which simply means that *the same data is available at multiple physical locations* [Ran10], is of great importance. The replicated data is also referred to as replicas. Being able to replicate data has two major advantages [Pla06]:

- Users can access the data as well locally as remotely which on the one side leads to a better fault-tolerance where problems are hidden from the user because the user can just address another copy if the local copy can not be accessed. This in turn increases the availability.
- Since users can access data locally and more bandwidth is available the performance can be increased as well.

Admittedly, *these advantages do not come without a cost.*[6] In order to introduce replicas an enterprise needs more storage and without efficient replication strategies the management of the different replicas will need a lot of computing and the error rate caused through inconsistency is very likely to rise.
Chapters 4.3.2 and 5 will allude the topic of replication strategies in more detail. Subsequently we will introduce a few space based computing implementations which address the replication of data and finally we will introduce an application scenario in more detail.
[Gar02] introduces an extended version of TSpace, called Enterprise TSpaces which aims to enhance fault-tolerance. ETS offers the possibility to choose between different replication possibilities in order to adress different application scenarios in the best possible way. Fault-tolerance in Linda was already a topic of interest in 1995 when [BS95] published an article on *supporting fault-tolerant parallel programming in Linda* and introduced FT-Linda. FT-Linda allows different recovery processes after a failure through specific transactions [RCVS05]. However, this

[6] `http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.admin.doc/admin622.htm`

solution is not ideal because tasks like removing faulty replicas have to be done specifically by the programmers.
Other space based computing implementations focusing on enhancing fault-tolerance through replication are amongst others DEPSPACE [BACF08], JADE [Li04], Linearizable Byzantine fault-tolerant Tuple Space [NBCdSFCL07] and [Bes06], S/Net Linda Kernel [CG86] and TIBCO ActiveSpaces[7].
The last paragraph strengthens the impression that data replication typically aims to improve *fault-tolerance or access time to tuples, while preserving a consistent view of the tuple space.*[8]
The application scenario, which will be presented next, has a slightly different aim and can be seen as an approach combining replication and mobility.
In chapter 2.3.3 we already have introduced LIME which is a space based computing implementation for MANETs and based on the ideas of Linda. [MP06] now enhances the classical LIME implementation with replication strategies in order to improve *availability in MANETs among hosts.*[9] Since hosts are not always connected to the federated tuple space it is desirable to still have certain information available when needed. The application scenario has been first presented in [MP04] and is called Tuling. Tuling is an application which aims to support the exploration of geographical areas, e.g. after a natural catastrophy[10]. Mobile users can explore the area and share location based information - e.g. pictures - with others through the esablishment of near-time wireless connections. The medium for doing so can be all sorts of mobile devices. Sharing Information efficiently in mobile ad hoc networks is not a trivial problem let alone introducing replicas - since *replication happens on the application level therefore does not enable further sharing of the information acquired.*[11] In order to make this possible an additional layer has been built on top of the LIME middleware and now allows to replicate data to different hosts. The replication process is illustrated in figure 3.2[12].
By adding replication to LIME, information can be made available even though a host is not connected to the federated tuple space. Compared to other application scenarios the focus at Tuling was less put on the consistency of replicas but more on making replication possible in the first place.

[7] `https://www.tibcommunity.com/blogs/activespaces`
[8] [MP06] page 17.
[9] [MP06] page 1.
[10] Chapter 3.4 will present another application scenario in the context of mobility which focuses on the support in emergency areas.
[11] [MP06] page 7.
[12] [MP06] page 8.

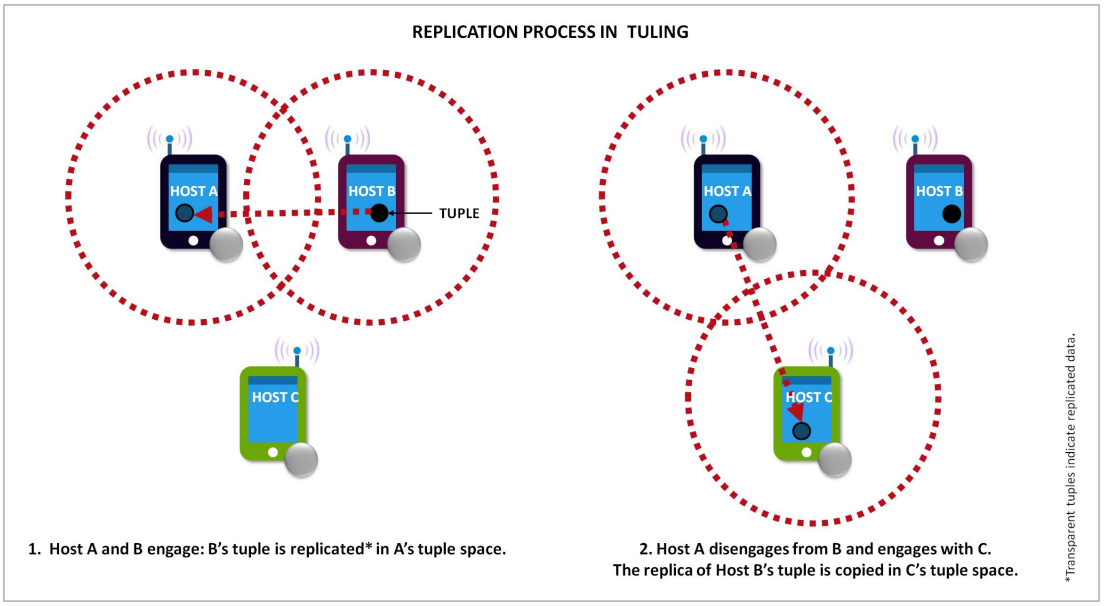


Figure 3.2: Replication Process in Tuling

3.3 Scenarios focusing on information sharing

Information is a valuable resource. In 2016 the information we are confronted with on an everyday basis can probably be compared to *bulleting down a super-highway.*[13] Therefore the process of information sharing has gained dramatically in importance. And it is no longer about writing emails, participating in conference calls or posting things on facebook. It's about finding, organizing, streamlining all the information we need in our private and professional life with the aim to make the most out of it.
On the professional level information sharing is a big topic in many areas just think about all the solutions focusing on collaboration, file sharing or optimizing processes in a supply chain. Managing information intelligently can even create business advantages. The application scenarios can range from strictly business focused solutions to military applications or social networks. Neglecting information can cause all sort of problems. Just imagine if you have to make decisions in realtime, and are not sure if you have up-to-date information. Bottom line the way information is shared and structured can decide between success or failure.
Such a time and safety critical application scenario is presented in [Mor10]. The SWIS[14] project[15] was created with the aim to guarantee high-availability and correctness of information in the aviation sector [Bar10] where a big number of inhomogeneous services [MMMB09] have to interact with each other in a coordinated way and new services can be integrated more

[13] `http://www.huffingtonpost.com/jalees-rehman/do-we-need-another-inform_b_1777985.html`
[14] System Wide Information Sharing
[15] The SWIS project was developed fom Frequentis AG, Austro Control, and the Institute of Information and Software Engineering Group at the Vienna University of Technology.

easily to the network. The project was realized with a XVSM space which helped to strenghten the coordination between the different services and enhance the fault tolerance of the system.
Another application scenario which will gain a lot of importance in the near and distant future can be found in the healtcare sector where clinical and administrative information needs to be shared effectively within the same and across different entities [For08]. As [For08] identifies in its study, healthcare organizations are willing to invest more money in clinical systems in order *to improve patient care and increase access to critical patient information.*[16]
Figure 3.3[17] illustrates different information sharing types between the different organizations, enterprises and care settings. It is important to understand that different types of information sharing pose different challenges. It is comprehensible that exchanging data between different stations of the same hospital is easier than exchanging data between regional health information organizations[18], e.g. a group of hospitals.

Organization / Care Setting	Within the Same Organization or Enterprise	Across Organizations or Enterprises
Within the Same Care Setting	**Type 1** **System to System Interfaces** Example: Lab, radiology, pharmacy interfaces in a hospital	**Type 2** **Limited Use RHIOs** Example: Hospitals sharing information with each other
Across Care Settings	**Type 3** **Enterprise Information Sharing** Example: Sharing information from an ambulatory practice to a hospital	**Type 4** **Comprehensive RHIOs** Example: Health care organizations sharing encounter information with each other

Figure 3.3: Different Information Types Across Organizations and Care Settings

The following application scenario was presented in [For08] and shows that by optimzing the information sharing process the quality of care and patient safety can be enhanced while costs can be reduced. *The Ann Arbor Area Health Information Exchange is comprised of four primary*

[16] [For08] page 18.
[17] [For08] page 4.
[18] RHIO

care and speciality practices[19] which minister about 400.000 patients and therefore have established a community health portal which automatically gathers the information about the patient - e.g. patient demographic information, medications, allergies, and current problem - into a coordinated clinical record. Through sharing this information with each other all practices get a better and more complete view on the patient.
Also space based computing implemations are focusing on application scenarios within the e-health sector, e.g. [NOVS11] and [CVF+07]. Another important fact is that when looking at e-health scenarios also topics like mobilty and security are of importance and need to be taken into consideration.

3.4 Scenarios focusing on mobility

People now more than ever want to enjoy the benefits of their computers - anywhere, at any time and without any bothersome wires. Mobile computing has not only become a big part of our private lifes it also has gained dramatically in importance in the corporate sector. Looking at the success of smartphones and tablet computers it is easy to see that mobile computing is more than just a trend.
While mobile computing is an exciting discipline with a lot of possibilities and good progess is made continuously [Jon12] there are still a lot of challenges that programmers face in this specific research area [Sat96]. Mobile elements often do not offer enough space to be resource-effective and they strongly depend on *finite energy sources.*[20]
When reading about mobile computing one often comes across terms like ubiquitous computing, nomadic computing, pervasive computing, mobile devices, mobile applications, wireless networks and mobile ad hoc networking [Ber04]. Each of these terms represents a concept or technology within the field of mobile computing. Before a few of the countless application scenarios are presented, it seems wise to explain the most important terms and definitions within the context of mobile computing.

- ***Ubiquitous computing is*** *the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.*[21]
- **Pervasive computing** differs from ubiquitous computing in the sense that pervasive computing integrates devices like mobile or smartphones in our daily life but does not want to hide their presence.
- **Nomadic computing** refers to the system support needed to provide all sorts of computing services to the mobile user who can move freely in his/her environment, e.g. home, work, public [BCKP95].
- **Mobile devices** allow people to access data, services and information from where ever they are, e.g. mobile phones and portable devices.

[19] [For08] page 8.
[20] [Sat96] page 1.
[21] [Wei93] page 1.

- **Mobile applications** are better known as mobile apps and represent differnt types of application software designed to run on mobile devices. Mobile applications have started out *as individual smaller software units with limited functions*[22] due to the limited hardware resources available on early mobile devices. Today apps already are a big part of our every day life and offer various services in all sectors, e.g. work, education, entertainment and networking.

- **Wireless networks** are all computer networks that are not connected by cables. Wireless networks transport information between the various computer devices with the help of radio waves which makes it possible to read e-mails, browse web pages, connect to a business network, see videos, talk with friends on social networks or connect to a mobile app regardless of your location. One can distinguish between WLANs[23], WPANS[24], WMANS[25] and WWANS [26].

- **Mobile ad hoc networks**[27] as described in [Per08], [Joh94] are wireless, mobile networks that do not need any fixed or preexisting infrastructure, e.g the Internet. All the administration and support normally performed by a conventional network is carried by the partizipating mobile nodes of a temporarilly formed ad hoc network.

- In the area of mobile computing **nodes** can be described as devices that are capable of connecting to a network from different entry-points.

Application Scenarios. There is a huge number of areas in which mobile computing applications can add value to our work, enhance the flow of information, or solve application scenarios that without mobility could not be realized. Application areas can be found amongst others in the arceological field [ADG99], retailing [NPS03], tourism [BC03] and in industry[28].
Mobile computing also offers a lot of interesting application scenarios for emergency services and transport. Subsequently two application scenarios realized with the help of space based computing implementations will be presented in more detail.
Emergency services. In [MQZ06] the reader is introduced to a first aid response scenario after a natural disaster, e.g. an earthquake. In such an environment wired or wireless networks are most likely only conditionally operational. Therefore [MQZ06] proposes to use RFID[29] tags to overcome this hindrance. The RFID tags could help to mark already explored areas, provide information about the accessibility for ambulance cars, get aware of injured people and treat them efficiently or send important information to the hospital before the patients arrive there. Figure 3.5 [MQZ06] illustrates a possible application first aid scenario for patients using RFID

[22] http://www.techopedia.com/definition/2953/mobile-application-mobile-app
[23] Wireless Local Area Network
[24] Wireless Personal Area Network
[25] Wireless Metropolitan Area Network
[26] Wireless Wide Area Network
[27] MANETs
[28] http://www.mobilise-europe.mobi/fileadmin/user_upload/EMMIA_PLP_1st_Summary_Report/EMMIA__PLP__wrap_up_BD_jv_21_06_12_KDK_at_2762012.pdf
[29] Radio Frequency Identification

tags and tuple spaces. In the scenario overview two patients are in the area. Patient A is already RFID tagged by a first aid rescuer and waits for a doctor to examine him. The RFID tag includes information which is important for the following process steps, e.g. the type of injury or the type of transport. In this example patient B is not registered by a first aid rescuer at the beginning but in the second process step this happens and she gets an RFID tag which will tell doctors that this patient needs examination. In the second process step patient A is examined by a doctor who writes new information to the RFID tag, namely that patient A has a broken leg and that transportation is needed and that ambulances can access this area. In the last of this three process steps one can see that patient A is in the ambulance now and will be transported to the next hospital or health center. The status is not closed because patient A will be cleared in the hospital once the treatment is finished. Patient B has more luck and is just under a small shock, the doctor clears her immediately and her status is closed.
This example showed that RFID tags can add real value to such an application scenario and *enforce pervasive tuple-based coordination activities.*[30] Nevertheless there are still issues like privacy, security or realibility which need to be adressed in the future.
There are several application scenarios focusing on emergency servicses, e.g. [CMMP] focuses on emergency control in buildings, e.g. in the case of fire or [LKHK05] introducing a mobile patient assistance system for adolescents with cancer.

[30] [MQZ06] page 6.

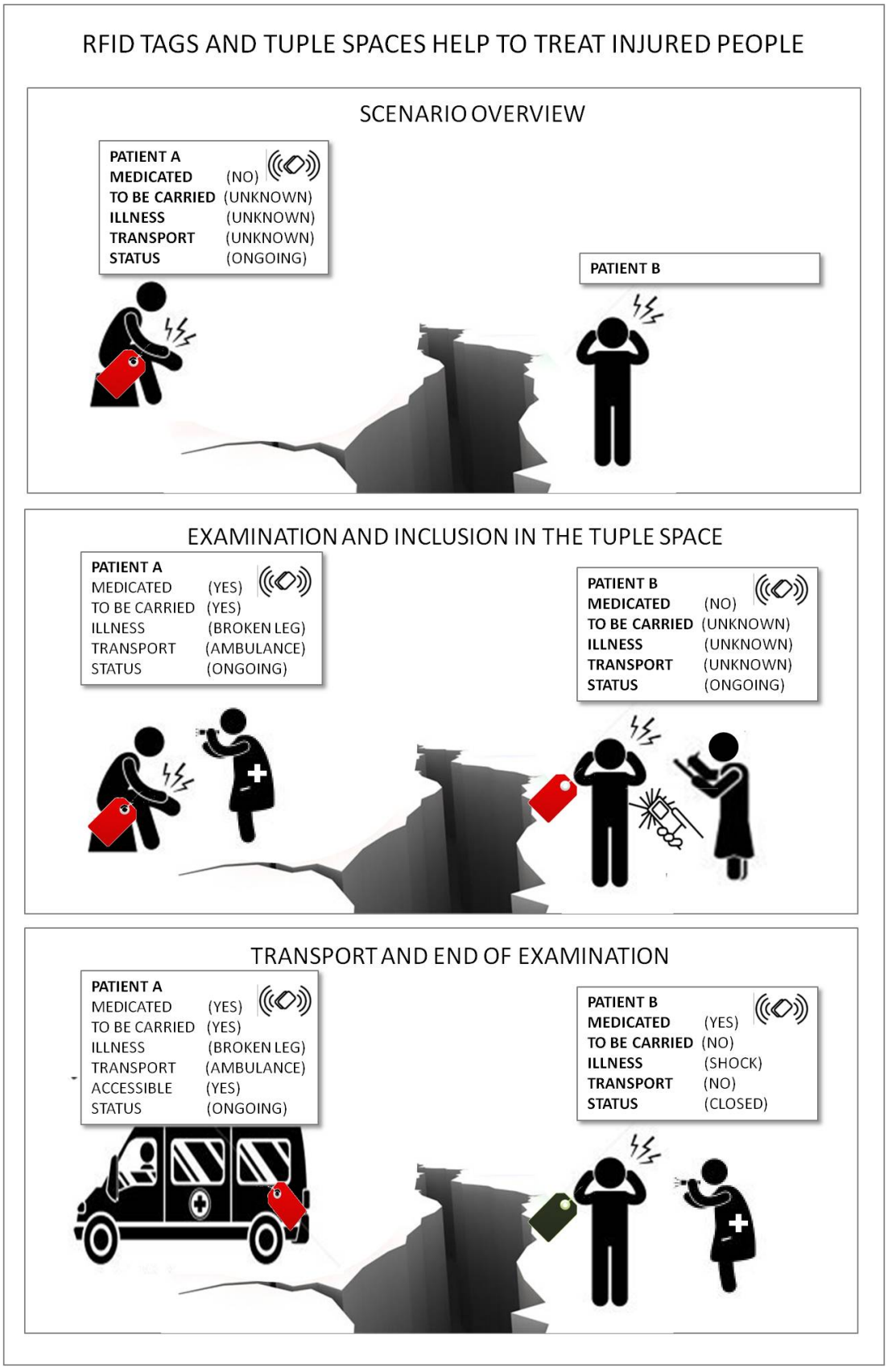


Figure 3.4: RFID Tags and Tuple Spaces Help to Treat Injured People Efficiently

Transport services. *Transportation and field services*[31] can be seen as key sectors of mobile computing. Countries all over the world and an armada of enterprises are exploring ITS[32] technologies to solve all sorts of traffic problems [SHBHDR11]. Topics like fleet management and asset monitoring, traffic monitoring, collision avoidance systems, traffic signal control systems, parking availability systems or traffic enforcement cameras are of special interst because they can smooth traffic operations, enhance security and most importantly save money.
An interesting application scenario in this field is the RealSafe[33] project which aims *to provide real-time information to car and driver*[34] with the overall goal to make roads safer and their use more efficient. Therefore the streets are equipped with so-called Road Side Units[35] which are connected in a *mesh wired broadband network*[36] and share informations within their network but most importantly with the cars on the road, e.g. information on traffic jams, road works or accidents. In this project XVSM and Distributed Hash Tables[37] were used in one middleware to enhance the coordination in the network [Bar10].
Another application scenario focusing on transportation is presented in [ZBS09] and falls into the category of traveler information systems. Travelers can connect to ATIS[38] and can request relevant information for their journey. Agents in the system hold the relevant information, e.g. traffic jams, delayed trains etc., which they either give after a specific request or send it actively to the traveler. This type of communication is not easy since its asynchronous and the agents do not know what kind of information the travelers will ask for. This application scenario is e.g. realized with LACIOS [39] which extends Linda and is a data-oriented coordination language which focuses on the design and implementation of multi agent systems. In LACIOS agents can publish or update their status and even condition their interaction with the environment which is of great use in transportation applications like presented above.

3.5 Scenarios focusing on security

Albert Einstein noticed once that *the world is a dangerous place to live.*[40][41] In the case of distributed systems this is even more true considering the huge amount of threats vulnerable distributed systems have to face. Hosts can be attacked by malware, trojan horses, spyware, worms and viruses. Plus hosts have to struggle with topics like eavesdropping, masquerading, message tampering, resource starvation and all sorts of unauthorized access issues. And as if

[31] `http://developer.att.com/home/community/conference/Tablets.pdf`
[32] Intelligent Transportation Services
[33] `http://www.ftw.at/projects/realsafe`
[34] [Bar10] page 18.
[35] RSU
[36] [BFK+11] page 5.
[37] DHT
[38] Agent Traveler Information Server
[39] Language for Agent Contextual Interaction in Open Systems
[40] `http://www.brainyquote.com/quotes/quotes/a/alberteins143096.html`
[41] Full Quote: The world is a dangerous place to live; not because of the people who are evil, but because of the people who don't do anything about it.

this would not be enough, threats can occur on infrastructure-, application- and service-level as well.
Under these aspects it is only normal that developers must adress such security issues when they develop distributed systems. A few well known security terms in distributed systems are explained subsequently [BCP+]:

- **Authentication** is the process of confirming the identity of a user/agent/computer/entity.
- **Authorization** helps to define the type of access a user/agent/computer/entity is granted. Certain operations or the system parts which need to be accessed are denied or permitted.
- **Availability** can be defined as the knowledge that information is available to authorized users/agents/computers/entities at any given time.
- **Confidentiality** refers to the process that restricts access to confidential information. It is crucial that information is shared only among authorised persons/entities. A good example is not to tell anyone the login information to your bank account.
- ***Data Integrity*** *refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle.*[42]
- **Privacy** can be seen as an *umbrella term*[43] which guarantees any user/agent/computer/entity the right *to keep information about themselves from others, revealing selectively.*[44]

One sector which is especially reliant on security in distributed systems is the financial sector. An application scenario is presented in [HJP02] presenting *a (simplified) global banking system.*[45] The banking system is realized through multiple JavaSpaces and one goal is to enhance the integrity of the system, e.g. ensuring that no money leaks from the system or that transfers are executed correctly and completely. In order to do so [HJP02] has experimented with *multi-phase locking and optimistic concurrency control.*[46]
Other application scenarios are often related to the topic of mobility which was presented in section 3.4. Section 3.4 has pointed out that application scenarios focusing on mobility are already a big part of our every day life and that they add value to it. A survey by Dimensional Research[47] showed that more and more people connect not only company owned but also their personal mobile devices to corporate networks [Res12]. Figure 3.6[48] displays this ratio and shows that 65 percent of all enterprises allow their employees to connect both personal and company owned mobile devices to the corporate network. Among those companies which allow personal mobile devices a dramatic rise can be noted. *78 percent saying more than twice as*

[42] http://en.wikipedia.org/wiki/Data_integrity
[43] http://www.it.cornell.edu/policies/infoprivacy/definition.cfm
[44] [BCP+] page 46.
[45] [HJP02] page 3.
[46] [HJP02] page 10.
[47] http://www.dimensionalresearch.com/
[48] [Res12] page 2.

many personal devices are used at work compared to two years ago.[49] This numbers are also displayed in Figure 3.6.

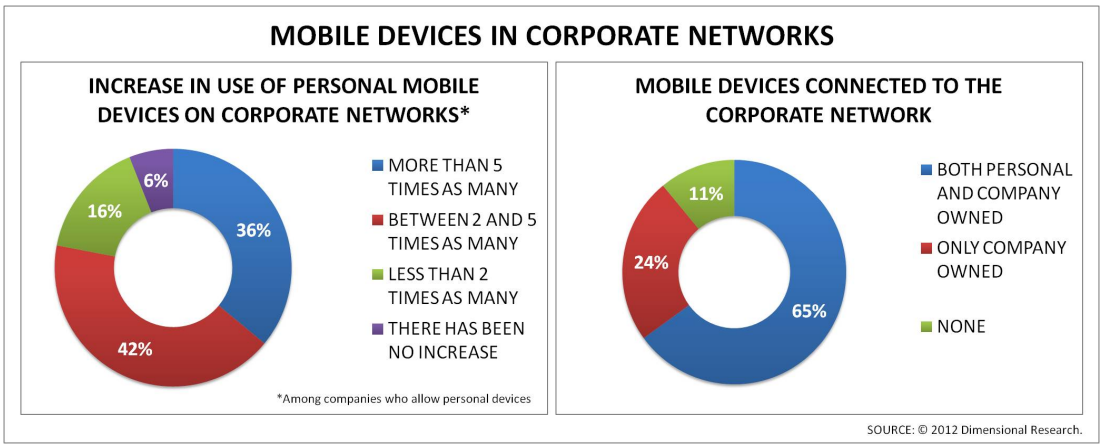


Figure 3.5: Mobile Devices in Corporate Networks

Bearing in mind that many mobile devices store sensitive customer and business data this is a worrying trend. McAffe[50] stated that mobile platforms have become increasingly attractive to cybercriminals as people extensively use smartphones and tablets, often without being aware of the risks. Another problem stressed in McAffes' report is the fact that malicious apps *are often integrated into trusted sources such as Google Play.*[51] So one could argue that one of the biggest threats to security is caused by the user her/himself.

Another lack of security in the context of mobility is certainly caused by the mobile agents themselves. Since we can find mobile agents in a great number of distributed application scenarios [HB01], e.g. E-commerce, parallel processing or entertainment, it is important to understand its impact on security. Although mobile agents show good perfomance and can execute certain tasks automatically [HB01], they can wreak havoc, if they are malicious.Which means that hosts and agents can be equally hurt.

Two interesting application scenarios have been presented by Lorenzo Bettini in [BDN02] and [Bet05]. Both implementations are based on Klava which is written in Java. The speciality of both application scenarios is the fact that they use *cryptographic primitives that enable encryption and decryption of tuple fields.*[52] The first application scenario presented is an encrypted chat system which allows to exchange encrypted messages, if wanted to [BDN02]. The second application scenario encrypts the data which the agent carries and therefore protects the integrity of the mobile agent. This allows the agent to travel to different sites and collect data for its owner [Bet05]. The agent uses a public key to encrypt all its data while it is roaming from site to site and once finished its owner can use the private key to decrypt the data.

[49] [Res12] page 2.
[50] `http://www.mcafee.com`
[51] `http://www.arnnet.com.au/article/454220/mobile_platforms_attractive_cybercriminals_mcafee_/`
[52] [BDN02] page 1.

Of course, there are other application scenarios focusing on different security features like protection mechanisms in order to enter a tuple space or protecting tuple fields from malicious attacks. An example for an ad hoc application scenario where the tuple space is protected through a password is given in [HR03] with a wireless dashboard application that allows drivers to carry out electronic payments at a tollbooth.
To sum it up security is a key feature of almost every distributed system and therefore the application scenarios can be found in a lot of areas. Chapters 4.3.5 and 4.9 will look in more detail into this topic in context with space based computing implementations.

CHAPTER 4

Classification of space based computing systems

4.1 Classification methodology

4.1.1 Classification Challenges

Chapters 2 and 3 have already shown that the number of space based computing systems and their diversity make it hard to draw an overall picture. The classification process itself posed two major challenges for me:

- The first hurdle which needs to be taken is the division of the entire set of space based computing systems to reasonable subsets which can be actually compared with each other. This is not completely straight forward since every space based computing systems has different focus areas and does not fit into every classification scheme. On the other hand it is important to look at the totality of space computing systems in order to draw the overall picture.
- The second and probably bigger challenge for me was to define suitable classification criteria for the space based computing systems. Here, various considerations have to be taken into account to develop a reasonable criteria catalogue, which is applicable to the space based computing systems but still offers the granularity to elaborate the distinctive features of every space based computing system. For example, [Sch08a] already stated that in the context of space based computing systems it is wise to choose classification criteria which allow to classify systems without an actual implementation since this is an often found phenomenon in the academic environment.

4.1.2 Classification methodology and criteria catalogue

The first step is to establish a criteria catalogue which defines suitable criteria for this survey. Good reference points for suitable criteria can be found amongst others in [ATS04], [CTZ02], [NSKMR08], [EFGK03] and [Sch08a]. Subsequently the chosen classification criteria is presented whilst giving a short overview of the structure of this chapter.
For the classification process of the various space based computing systems a top-down approach has been chosen, which starts with a high-level classification after families in section 4.2.
After that I will go one level deeper and have a look at the operations in section 4.3 which space based computing systems can support. In chapter 2 a few systems have been already presented in more detail and in this context basic operations as well as the general ideas of notifications, transactions have been introduced to a certain extent. To complete the picture I will also write about flow control and its importance for space based computing systems.
Section 4.4 will focus on the topic of coordination types. A wide range of coordination types, e.g. Linda coordination, coordination with lists, arrays or hashtables will be used as criteria to survey the space based computing systems.
Section 4.5 and 4.6 round off the picture which we started to paint in section 4.4 addressing the topics of how the data within the space can be structured and what types of data can be stored in the space.
The next group focuses on qualitative criterias. The decision to put the topics of extensibility, security and life cycle management in one group was made by me because I believe that these topics are hard to measure but can influence thw quality of a space based computing systems.
In section 4.7 spaces are classified after their extensebility. This is a relatively important criteria since it defines how flexible a space based computing system is and how easy it is customisable to the wishes and needs of developers and application scenarios.
Section 4.8 and 4.9 focus on two very specific but absolutely relevant topics, namely on security and life cycle management. Focusing on security the space based computing systems will be classified after their authentication and authorization mechanisms. Also the topics of data encryption and data signing are elaborated. In section 4.9 I will discuss the topic of garbage collection in context of space based computing systems.
Both scalabilty and and performance can be seen as quantitative indicators. Since quite a few space based computing systems only exist on paper it makes no sense to classify them all. We will discuss a quantitative approach presented by [Lwe10] but also focus on fault-tolerance and availability in the context of performance.

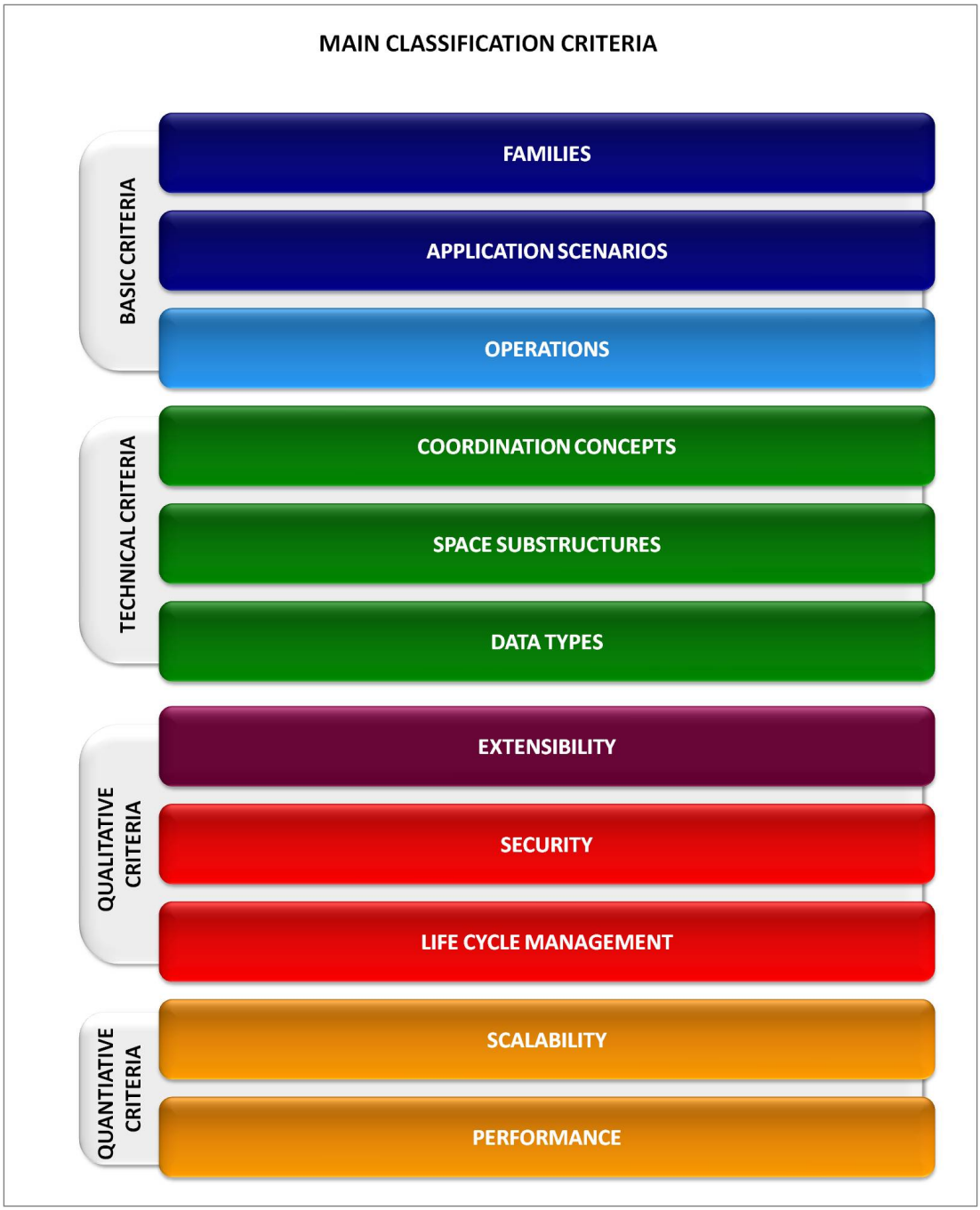


Figure 4.1: Criteria Catalogue for Classification of Space Based Computing Implementations

Figure 4.1 gives a visual overview of the main criteria catalogue. similar color schemes represent similar areas of criteria. Classification after families, application scenarios[1] and operations can

[1]Classification after application scenarios was already discussed in detail in chapter 3 - a detailed overview can be

be seen as basic classification criteria which are a good background to understand the ideas of space based computing systems.
When looking at a certain criterion I will always start with the complete set of space based computing systems and discuss the main issues a space based computing system has to fulfill to meet the specific criterion. Thereby we immidiately generate subsets: a set which fullfills the criterion completely, partly or not at all. For further discussion I only will take those subsets into account which fullfill the criterion at least partly. However, in the summary for each criteria section the specific criterion matrix for all space based computing systems is presented.
Chapter 5, the conclusion will present the entire criteria matrix in order to give a complete overview.

4.2 Classification by family

Before starting with the classification process a short description of the used methodology is given. In order to classify space based computing systems by family one has to understand that they are influenced by different visions and paradigms. In the case of space based computing systems three paradigms are worth mentioning:

- The semantic web paradigm.
- The space based computing paradigm.
- The P2P[2] and grid computing paradigm.

Since the vision of space based computing has already been discussed in more detail in chapters 2 and 3, only a short summary will be given and then the focus will be put on the semantic web as well as on P2P and grid computing.
In a next step I will discuss main technologies and standards in this field, how they interact with each other and which paradigms they follow. In this context Linda, Java spaces, JXTA[3], and XVSM will be discussed amongst others.
In a last step I will put the space based computing systems into context with the presented paradigms and technology standards.
Figure 4.2[4] represents the result of the classification process of space based computing systems by family. To show the results upfront will help to align the information given subsequently with the overall picture.

found in the entire criteria matrix in chapter 5.
[2]Peer-to-Peer
[3]Juxtapose
[4]Figure 4.2 is partly influenced from `http://www.complang.tuwien.ac.at/eva/fileadmin/user_upload/teaching_material/SBC/SS2010/Teil-I.pdf` slide 29.

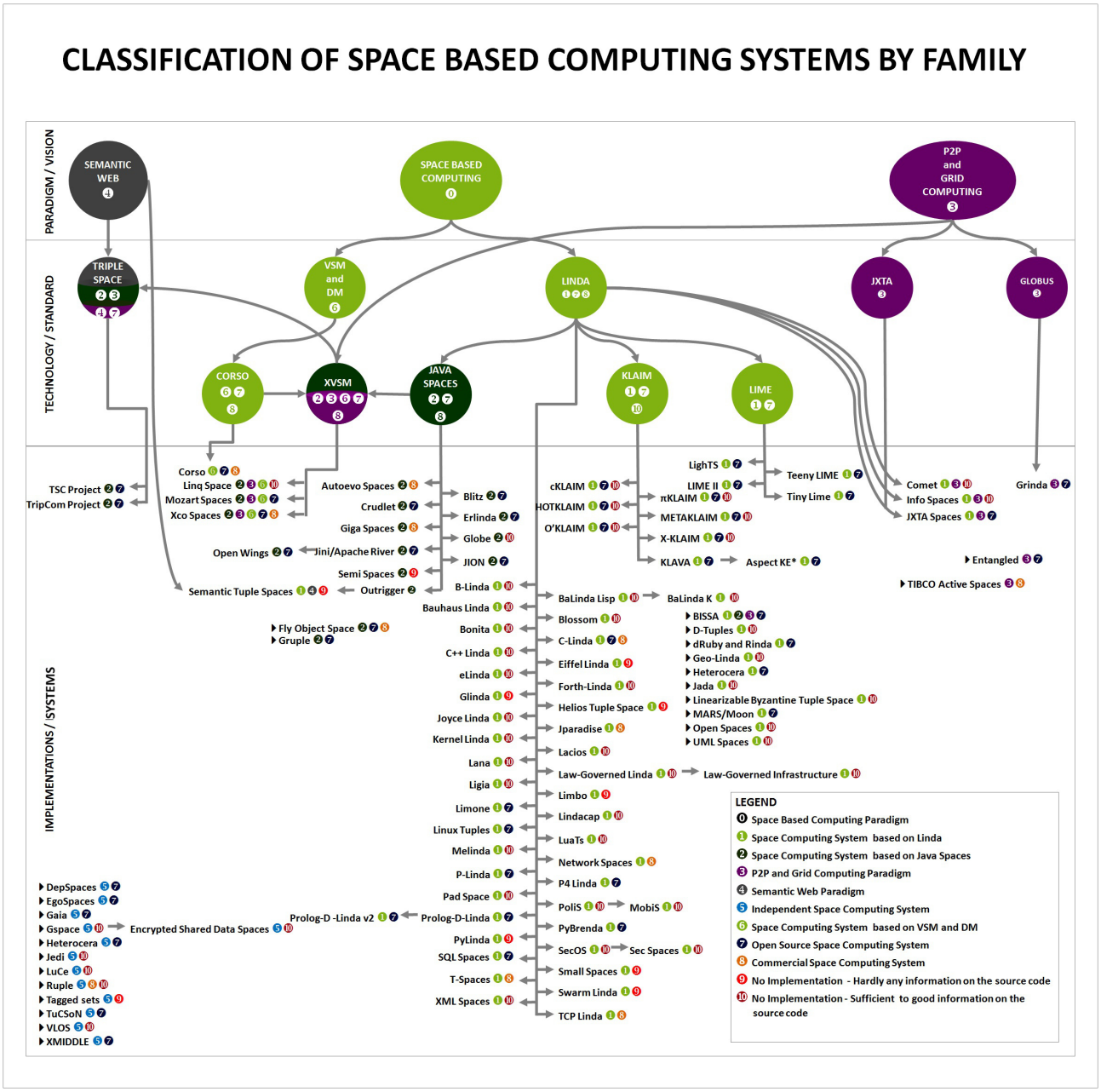


Figure 4.2: Classification of Space Based Computing Systems by Family

4.2.1 Paradigms and visions

Looking at figure 4.2 immediately will make you notice one thing: The systems mainly influenced by the pure space base computing paradigm dominate the picture.

The P2P and grid computing paradigm

Peer-to-Peer. The term P2P was coined in the year 2000 [Ora01]. A music-sharing application called Napster showed that Internet has more to offer than the client/server model[5]. So Napster can be seen as the mother of the P2P trend although it is not a pure P2P system[6], since a central database server manages all incoming queries [SF02]. As soon as a peer logs into the Napster network, the server registeres all the files the user wants to share with other peers. Search queries return a list of peers that fullfill the search requirements. The user than can establish a direct connection with one of the peers and download the desired file. Figure 4.3 illustrates how Napster works.

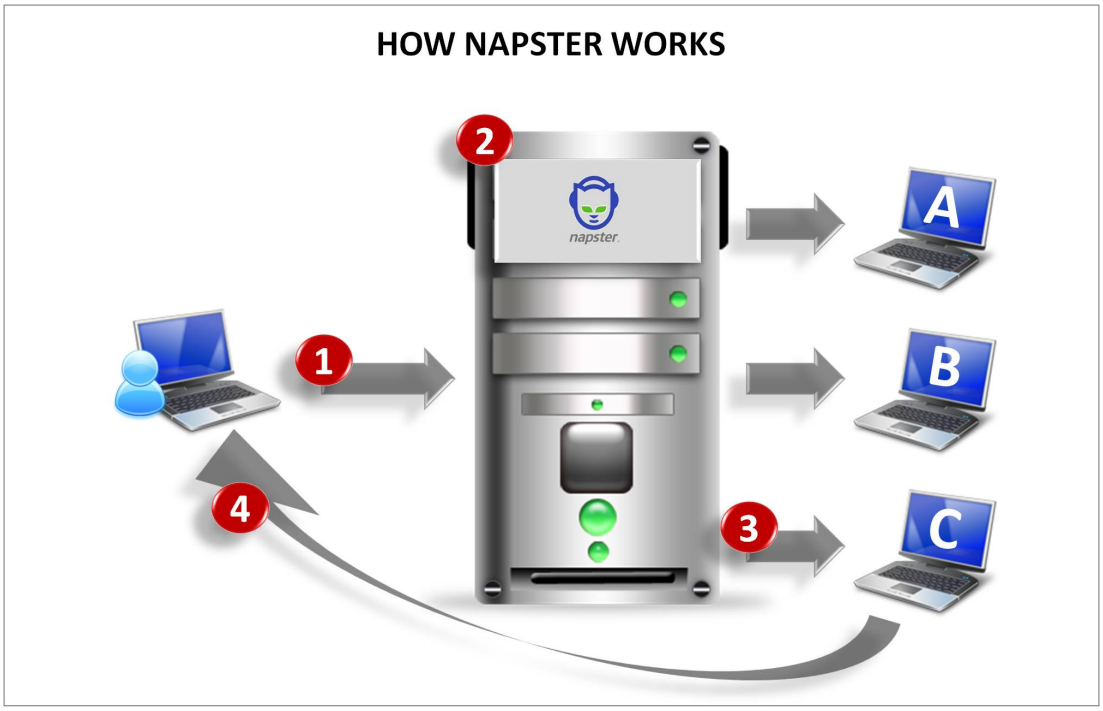


Figure 4.3: How Napster Works

The workflow represented in figure 4.3 describes the four major steps in the operating mode of Napster[7]:

1. A user that is connected to the Napster network sends a request for specific file (e.g. a song).

2. The Napster server checks with its database if the requested file is available on another peer in the network.

[5]Funny enough, the Internet was designed as a P2P system initially.
[6]Napster is more an extension of the classical client/server model.
[7]`http://ntrg.cs.tcd.ie/undergrad/4ba2.01/group10/napster.html`

3. The in figure 4.3 requested file is found on the peer called C.
4. The user and peer C can now establish a direct connection where peer C sends the requested file without further detours to the user.

It's easy to see that Napster offers quite a few advantages[8]:

- An easy to control structure.
- The big amount of data is handled by the peers and not the server.

These advantages are accompanied by the subsequently described disadvantages:

- The server represents the single point of failure.
- Private data is stored on an external server[9].
- The Napster server can not scale.

After presenting Napster I will talk about real P2P systems in general. [MKL+03] defined P2P as follows:
The term "peer-to-peer" refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner.[10]
P2P networks can be used for different application scenarios [Ji04], e.g. file sharing, collaboration or distributed computing amongst others. P2P models can be classified into two catogeries, namely structured or unstructured P2P models [HD05].
Peers in unstructured P2P systems do not hold any information about the resources of other peers which makes a well-directed query impossible. Later examples will show that in unstructured P2P systems a request is forwarded as long as a suitable match is found or the lifespan of the query expires. Peers in structured P2P systems have knowledge of the resources of other peers which allows a target-orientated search. However managing this additional information leads to higher operating expenses.
Unstructured P2P models can be further divided according to their structure into centralised, pure and hybrid P2P models. Figure 4.4[11] illustrates the difference between these three types.

[8] `http://www.complang.tuwien.ac.at/eva/fileadmin/user_upload/teaching_material/SBC/SS2010/Teil-I.pdf` slide 21.
[9] In the year 2000 this was definitely seen as a drawback and one of the main reasons why Napster was sued.
[10] [MKL+03] page 1.
[11] Influenced by [Ji04] page 6.

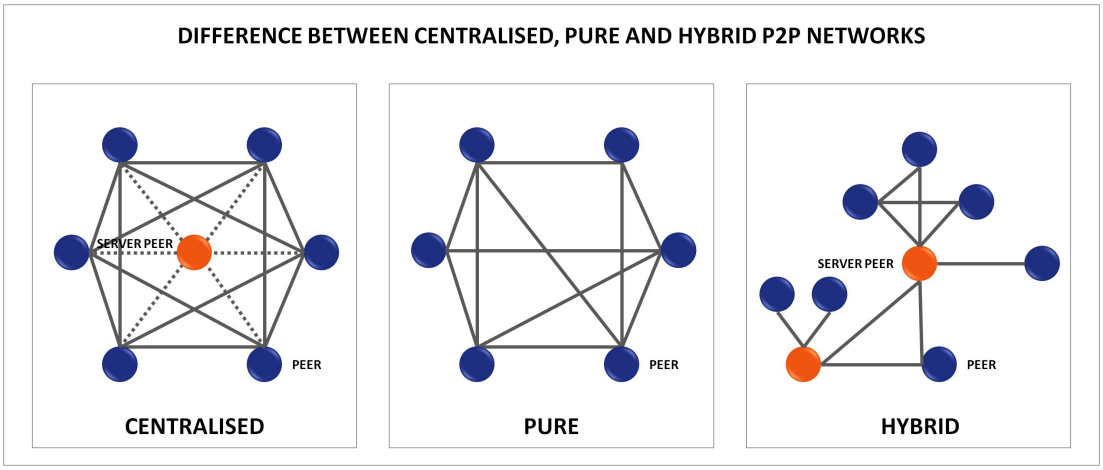


Figure 4.4: Difference between Centralised, Pure and Hybrid P2P Networks

They main difference between these three types is their routing behavior of queries [ESZK04]. Centalised P2P models like Napster need a centralised server in order to function. Hybrid P2P networks always includes dynamic server peers which function as central entities and therefore can optimize the routing behaviour. The advantage of hybrid over centralised P2P networks lies in the fact that the loss of one of the server peers does not bring the entire system to a halt. Pure P2P networks [Sch01] are missing such server peers and work completely without central entities. Examples for unstructured P2P systems are amongst others:

- I already discussed Napster which belongs to the family of centralised P2P systems. The main drawbacks of these P2P systems are that they represent a single point of failure and that they are vulnerable to performance bottlenecks. Another member in this family is for example BitTorrent[12].
- Gnutella 0.4[13] is a representative of a P2P network and it works completely without central entities. Therefore it is fault resistant but its unstructured network architecture can not assure that the requested information can be found in reasonable time which leads to a considerably high operating costs. Another risk can be seen in the fact that if one peer leaves the network the entire system could decompose into sections that can not communicate with each other. Freenet[14] also belongs in the category of P2P systems.
- Hybrid P2P systems like KaZaA[15] and Gnutella 0.6[16] are part of the hybrid P2P systems family. Here the server peers also called supernodes hold information about the resources of their children (leaf peers). It's obvious that searching in hybrid P2P systems is a lot easier than in pure P2P systems.

[12] http://www.bittorrent.com/
[13] http://en.wikipedia.org/wiki/Gnutella
[14] https://freenetproject.org/
[15] http://en.wikipedia.org/wiki/Kazaa
[16] http://en.wikipedia.org/wiki/Gnutella2

As written already structured P2P networks hold infomation about the resources of other peers. Most of the structured P2P systems like CAN[17] [RFH+01] or Chord[18] are based on distributed hash tables[19] [ESZK04]. DHT make it easy to find information, because only the appropriate key and value is needed to execute a target orientated search. Furthermore every peer has only to know about its assigned part in the hash table and a few distinctive neighbours[20] Disadvantages of structured P2P systems can be seen in the complicated management of peers which dynamically enter or leave the network and the fact that only exact matches can be found.

Grid Computing. Grid Computing can be seen as a way to unite resources from many computers within a network in order to solve a single problem[21]. The perfect grid computing scenario would turn a set of simple peers within a computer network into one supercomputer. Although grid computing itself is a huge topic to tackle, I will stick to a simple definition in order to reduce complexity. The three major pillars that define a grid are [Fos02]:

- *A grid must coordinate resources that are not subject to centralized control.*
- *A grid must use standard, open, general-purpose protocols and interfaces.*
- *A grid must deliver nontrivial qualities of service (e.g., relating to response time, throughput, availability, and security) for coallocating multiple resource types to meet complex user demands.*[22]

The qualities of the space based computing paradigm and the P2P and grid computing paradigm interlock really well and can enrich each other when it comes to topics like the efficient use of resources, coordination and search techniques.

The semantic web paradigm

A very good informal definition[23] of the term semantic web reads as follows: *The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*[24]
Until the article of Bernes-Lee, the World Wide Web[25] was seen as an extensive collection of documents which accommodates an unbelievable amount of information which can be only interpreted by human beings. Therefore the main goal was to create a web in which machines can interpret data and by adding semantics to it can actually create knowledge [Wei11].

[17] Content-Addressable Network
[18] `https://github.com/sit/dht/wiki`
[19] DHT
[20] `http://www.complang.tuwien.ac.at/eva/fileadmin/user_upload/teaching_material/SBC/SS2010/Teil-I.pdf` slide 25.
[21] `http://www.howstuffworks.com/grid-computing.htm`
[22] [Fos02] page 2-3.
[23] `http://www.w3.org/RDF/Metalog/docs/sw-easy`
[24] [BLHL01] page 34.
[25] WWW

In 2012 the Semantic Web is used in a big number of fields although it is still in its infancy. Nevertheless the semantic web paradigm can support the ideas of space based computing and vice-versa, which can be seen amongst others in [NAP+07], [CVF+07] and [NSKMR08]. At the moment the semantic web paradigm influences just a few space based computing systems but it is obvious that the more mature it becomes the more it will influence other areas.

4.2.2 Technology and standards

Figure 4.2 also shows that there are a few technologies and standards which are so dominat that most of the space based computing systems follow them in one or the other way. Looking at these technologies and standards will also show that one standard can combine more than one paradigm. In the following subsections I will introduce the most important standards and technologies shortly, if they have not been introduced in chapter 2.3 already. The easiest way to classify these technologies is to to relate them to the paradigms they follow.

Technology and standards following the space based computing paradigm

The main two standards in this field are Linda and Java Spaces, which is a direct derivative of Linda. Both space based computing systems were described in in chapter 2.3. Linda as the mother of space based computing is the main influencer for most space based computing implementations (52 of all 103 analysed space based computing systems are directly influenced by the ideas of Linda.). Java Spaces one of the many children of Linda also gained a lot of interest within the community and soon won several imitators due to its charm and close connection to the Java world. Two other standards which are directly influenced by Linda are LIME and KLAIM. Both, LIME and KLAIM pay particular attention to the topic of mobility, choosing different approaches to address certain problems in this context. LIME was already described in chapter 2.3 and KLAIM will be described in more detail in later parts of this chapter.
The ideas of a virtual shared memory can be compared to Linda, which also can be seen as an example of a virtual shared memory. That is also the reason why VSM is placed beside Linda in figure 4.2. The advantage of such a virtual shared memory approach is that diverse applications can interact with each other through a shared virtual data space. *The interaction is coordinated in a structured and secure way*[26] where finding the relevant data is possible without knowing its specific location. Although much research has been undertaken in this field, virtual shared memory solutions failed to reach a status where one would refer to them as standard [Kue98] but for the sake of completeness it is important to classify it. A derivative of such a virtual shared memory is Corso[27] which extends Linda through a virtual shared memory space of Java objects and has been presented in 2001 by the tecco Software Entwicklung AG, a spin-off of the Technical University of Vienna at the institute for computer languages. Corso offers the following features [Ang03] like transactional security, notifications, automatic garbage collection, authorization mechanisms, support for various computer languages and the possibility for parallel process execution.

[26] `http://www.complang.tuwien.ac.at/eva/SBC-Group/sbcGroupIndex.html`
[27] Coordinated Shared Objects

Corso can be seen as the predecessor of XVSM (also described in chapter 2.3) which is influenced not only by the space base computing paradigm but also by the P2P and grid computing paradigm as well as Java Spaces and Corso.

Technology and standards following the P2P and grid computing paradigm

In the context of P2P JXTA can be seen as a relevant standard[28]. JXTA itself is open source and initially developed by Sun Microsystems in 2001. JXTA is composed of peer-to-peer networking protocols that allow any connected device on the network - e.g. servers, PCs, PDAs or mobile phones - to communicate with each other. Another advantage is that JXTA supports various languages, operating systems, hardware and transport protocols due to the fact that it is based on protocols[29]. This flexibility can be quite interesting for programmers. JXTA has received a lot of attention in the past but in November 2010, Oracle announced its withdrawal from the JXTA projects and since 2011 no information about the future of JXTA has been given. `http://jxta.kenai.com/` offers still information about JXTA.
The main standard in the context of grid computing is Globus[30] which was first introduced by the Globus Alliance in 1995. On the Globus website all information on the Globus Toolkit can be found. The Globus Toolkit allows to build grids for different application scenarios. The Globus Toolkit mainly acts as middleware in a grid network. It provides the infrastructure to install grid services, it coordinates the access to resources, it provides a security structure and specific protocols such as GridFTP [BO05].

Technology and standards following the semantic web paradigm

The Triple Space combines the ideas of the semantic web, XVSM and therefore P2P with each other. There are other space based computing systems which integrate the ideas of the semantic web. Such space based computing systems will be discussed later in chapter 4.

4.2.3 Implementations, products and systems in the context of space based computing systems

Looking at figure 4.2 one can divide between two kind of space based computing systems:

- Space based computing systems which are directly influenced by a certain technology or standard.
- Space based computing systems which can not be directly attached to one of the technologies or standards mentioned above but still can be classified as space based computing systems.

Most of the space based computing systems (about 76 percent of all analyzed space based computing systems) are directly influenced by the above presented standards and technologies. The

[28] `http://en.wikipedia.org/wiki/JXTA`
[29] `http://www.webopedia.com/TERM/J/JXTA.html`
[30] `http://www.globus.org`

other 24 percent can be seen as more or less independent and are marked with the number 5 in figure 4.2.

Space based computing systems mainly influenced by the space based computing paradigm

46 of all 103 space based computing systems are direct derivatives of Linda. Many of these systems extend the ideas of Linda simply with an another programming language, e.g. BaLinda Lisp [YW90], C-Linda [Gel85], C++ Linda [DF96a], Eiffel Linda [Jel90] or LuaTS [LR03] among others.
Other space based computing systems in this category have developed their own children, such as BaLindaLisp which can be seen as the parent of BaLinda K [YF96] or Law-Governed Linda [ML95] which is the predecessor of Law-Governed Infrastructure [MMU01].
Figure 4.2 also shows as the children of KLAIM and LIME. Both standards have a relevant number of children in 2015. The children of LIME - LighTS [PB05], LIME II [AAHC09], TeenyLIME [CMMP06] and TinyLIME [CGG+05c] - can be seen as the systems with a more practical approach since all of them posess full implementations whereas the children of KLAIM[31] [BBDN+03] - cKLAIM, HOTKLAIM[32], METAKLAIM or O'KLAIM amongst others - offer theoratical approaches since no source code is available.
Since Java Spaces can be seen as one of the most well known standards when it comes to space based computing it is no wonder that it has quite a lot of derivatives like Outrigger, GigaSpaces or Blitz amongst others. Most of the space based computing systems which are influenced by Java Space have full implementations. Under the direct Java Spaces derivatives only the Semantic Tuple Spaces [KLF04] combine the space based computing paradigm with the semantic web paradigm. The sTuples system extends Outrigger, the reference implementation of Java Spaces to support the intelligent matching process of tuples. One of the more famous commercial space based computing systems is GigaSpaces which still is very popular in 2015.
XVSM has been improved continously over the years and already has quite a few derivatives. Mainly one can to distinguish between two different implementations:

- MozartSpaces: This is the open source Java implementation, where documentation and the source can be found on the project website under `http://www.mozartspaces.org`.
- Xcoordination Application Space and Xcoordination Coordination Space: is the .NET version which exist as a commercial and open source version.

All children of XVSM have implementations which are continuously improved, e.g. [KRML08b] and information on it can be found on the project website[33] which offers also publications et cetera. One of the newer space based computing systems extending XVSM is called Linq Space and was presented by [Gel11].

[31] Kernel Language for Agents Interaction and Mobility
[32] Higher-Order Typed KLAIM
[33] `http://www.complang.tuwien.ac.at/eva/research/researchpublications.html`

Many of the space computing systems which are not directly linked to a certain standard or technology still follow the ideas of Linda, e.g. BISSA [WWF+10] which can be seen as a combination of the ideas of Linda, Java Spaces and P2P. Another interesting free space based computing system which can be related to the ideas of Linda is called Geo-Linda [PCBB07]. Geo-Linda focuses on a very specific application scenario which differentiates it from other space based computing systems. Geo-Linda addresses the problem of *detecting different kind of movement patterns of devices*[34], e.g. the insertion of a product in a shopping cart, the loading of containers in trucks or boats or helping visually impaired people to take the bus. One could see certain parallels to LIME but Geo-Linda uses different operations which sets it apart. Of course there are many other free space based computing implementations to discuss them all would go beyond the scope of this section. The main message when talking about free based computing implementations is that they either combine different technologies and paradigms or have different approaches and aims which makes it hard to relate them to just one standard or paradigm.

Space based computing systems mainly influenced by the P2P, grid computing and the semantic web paradigm

Figure 4.2 shows that only 4 space based computing systems can be found in the section mainly focusing on P2P and grid computing. When it comes to systems which are mainly influenced by P2P Comet [LP05], Info Spaces [BMSV] and JXTA Spaces [Li01] are the one to name - all three are based on JXTA with strong influences from Linda which also shows how the paradigms interlock with each other. In the context of Grid computing one has to name *Grinda*[35] [CM08b] which is a tuple space implementation for the Globus Toolkit 4 and can be used to coordinate distributed tasks without knowing host identities and network topology.
Space based computing systems directly influenced by the semantic web paradigm are even rarer to find. TripCom[36] combines technologies like tuple spaces, web services and the semantic web. Chapter 2.3.4 already discussed Triple Space Communication in more detail. [NSKMR08] presented two more space based computing systems in the context of the semantic web paradigm:

- Conceptual Spaces[37] which was meant to extend Triple Space Communication with more features.
- Semantic Web Spaces [TN04] is a *middleware for the Semantic Web, enabling clients using Semantic Web data to access and process knowledge to coordinate their interdependent activities.*[38]

Those two space based computing systems have not been part in the classification process since Conceptual Spaces can be seen as an extension to Triple Space Communication and Seman-

[34] [PCBB07] page 3.
[35] Grid + Linda
[36] Triple Space Communication
[37] CSpaces
[38] [NSKMR08] page 196.

tic Web Spaces can be seen as an extension of XML Spaces. Nevertheless when appropriate Semantic Web Spaces will be included in the classification process.

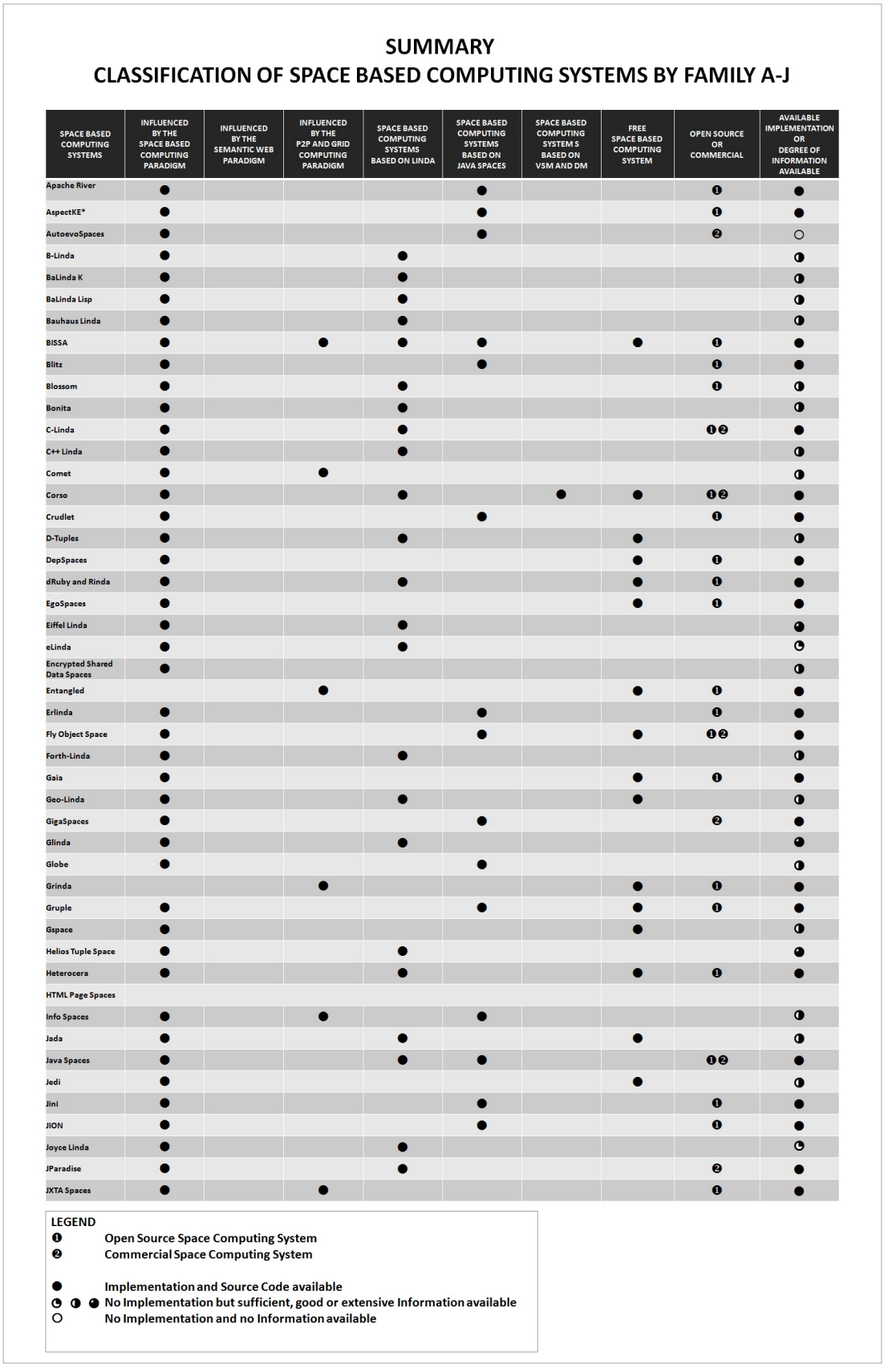
Independent space based computing systems

Independent space based computing systems can not be associated straight forward to Linda or any other of the before presented standards or technologies. Nevertheless they use the idea of a commonly shared space for their coordination. A representative space based computing system for the family of independent space based systems is TuCSon, which was introduced in chapter 2.3.5 earlier on. Although TuCSon *is a Java-based*[39] *model for the coordination of distributed processes, as well as autonomous, intelligent and mobile agents.*[40] and it shares a few basic operations with Linda but it is different in so many other ways that the word independent is probably the right one to describe it. One of this major differences is that TuCSoN offers programmable tuple centres , which can react to incoming or outgoing communication events[41]. The main message which the reader should bear in mind is that a space based computing system is classified as independent because it uses only a small notion of the common ideas and brings some completely different ideas or technologies to its approaches.

Summary

Two other aspects which were not discussed yet but also shown in figure 4.2 are if a space based computing system belongs to the commercial or open source family and if an implementation is available or not. In case no implementation is available the degree of the information on the source code given in papers is indicated. To find information on a single space based computing system faster figure 4.5 - figure 4.7, a summary matrix of the classification of space based computing systems by family, is introduced. Here again connections to paradigms and technologies are displayed as well as the information if a system belongs to the open source or commercial family. The information if a system has an implementation is given in 25 % steps displaying the different states for no implementation, sufficient, good, excellent information and implementation available. 0 % stand for no implementation and 100 % for implementation available.
An interesting aspect is that nearly 50 % of all space based computing systems actually have an implementation. From this 50 % percent only about 12 % belong only to the commercial family and 14 % offer an open source solution as well as a commercial one. 74 % of this 50 % are open source.

[39] also Prolog-based
[40] `http://apice.unibo.it/xwiki/bin/view/TuCSoN/`
[41] [NOVS11] page 4.



SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY FAMILY A-J

SPACE BASED COMPUTING SYSTEMS	INFLUENCED BY THE SPACE BASED COMPUTING PARADIGM	INFLUENCED BY THE SEMANTIC WEB PARADIGM	INFLUENCED BY THE P2P AND GRID COMPUTING PARADIGM	SPACE BASED COMPUTING SYSTEMS BASED ON LINDA	SPACE BASED COMPUTING SYSTEMS BASED ON JAVA SPACES	SPACE BASED COMPUTING SYSTEM S BASED ON VSM AND DM	FREE SPACE BASED COMPUTING SYSTEM	OPEN SOURCE OR COMMERCIAL	AVAILABLE IMPLEMENTATION OR DEGREE OF INFORMATION AVAILABLE
Apache River	●				●			❶	●
AspectKE*	●				●			❶	●
AutoevoSpaces	●				●			❷	○
B-Linda	●			●					◑
BaLinda K	●			●					◑
BaLinda Lisp	●			●					◑
Bauhaus Linda	●			●					◑
BISSA	●		●	●	●		●	❶	●
Blitz	●				●			❶	●
Blossom	●			●				❶	◑
Bonita	●			●					◑
C-Linda	●			●				❶❷	●
C++ Linda	●			●					◑
Comet	●		●						◑
Corso	●			●		●	●	❶❷	●
Crudlet	●				●			❶	●
D-Tuples	●			●			●		◑
DepSpaces	●						●	❶	●
dRuby and Rinda	●			●			●	❶	●
EgoSpaces	●						●	❶	●
Eiffel Linda	●			●					◕
eLinda	●			●					◔
Encrypted Shared Data Spaces	●								◑
Entangled			●				●	❶	●
Erlinda	●				●			❶	●
Fly Object Space	●				●		●	❶❷	●
Forth-Linda	●			●					◑
Gaia	●						●	❶	●
Geo-Linda	●			●			●		◑
GigaSpaces	●				●			❷	●
Glinda	●			●					◕
Globe	●				●				◑
Grinda			●				●	❶	●
Gruple	●				●		●	❶	●
Gspace	●						●		◑
Helios Tuple Space	●			●					◕
Heterocera	●			●			●	❶	●
HTML Page Spaces									
Info Spaces	●		●		●				◑
Jada	●			●			●		◑
Java Spaces	●			●	●			❶❷	●
Jedi	●						●		◑
Jini	●				●			❶	●
JION	●				●			❶	●
Joyce Linda	●			●					◔
JParadise	●			●				❷	●
JXTA Spaces	●		●					❶	●

LEGEND
❶ Open Source Space Computing System
❷ Commercial Space Computing System

● Implementation and Source Code available
◔ ◑ ◕ No Implementation but sufficient, good or extensive Information available
○ No Implementation and no Information available

Figure 4.5: Summary Classification of Space Based Computing Systems by Family A-J

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY FAMILY K-S

SPACE BASED COMPUTING SYSTEMS	INFLUENCED BY THE SPACE BASED COMPUTING PARADIGM	INFLUENCED BY THE SEMANTIC WEB PARADIGM	INFLUENCED BY THE P2P AND GRID COMPUTING PARADIGM	SPACE BASED COMPUTING SYSTEMS BASED ON LINDA	SPACE BASED COMPUTING SYSTEMS BASED ON JAVA SPACES	SPACE BASED COMPUTING SYSTEM S BASED ON VSM AND DM	FREE SPACE BASED COMPUTING SYSTEM	OPEN SOURCE OR COMMERCIAL	AVAILABLE IMPLEMENTATION OR DEGREE OF INFORMATION AVAILABLE
Kernel Linda	●			●					◑
Klava	●				●			❶	●
L²imbo	●			●					◕
Lacios	●			●					◑
Lana	●			●					◑
Law-Governed Infrasturcture	●			●					◑
Law-Governed Linda	●			●					◑
LighTS	●			●				❶	●
Ligia	●			●					◑
Limbo	●			●					◕
LIME	●			●				❶	●
LIME II	●			●				❶	◕
Limone	●			●				❶	●
Linda	●			●				❶❷	●
Lindacap	●			●					◑
Linearizable Byzantine Tuple Space	●			●			●		◑
LinqSpace	●		●		●	●			◑
Linux Tuples	●			●				❶	●
LuaTs	●			●					◑
LuCe	●			●			●		◑
MARS/Moon	●			●	●		●	❶	●
Melinda	●			●					◑
MobiS	●			●					◑
Network Spaces	●			●				❷	●
Open Spaces	●			●	●		●		◑
Open Wings	●				●			❶	●
P-Linda	●			●				❶	●
P4 Linda	●			●				❶	●
PadSpace	●			●					◑
PoliS	●			●					◑
Prolog-D-Linda v2	●			●				❶	●
PyBrenda	●			●				❶	●
PyLinda	●			●					○
Ruple	●						●	❷	◑
Semantic Tuple Spaces	●	●			●				◕
SecOS	●			●					◑
SecSpaces	●			●					◑
SemiSpace	●				●				○
SmallSpaces	●			●					○
SQLSpaces	●			●				❶	●
Swarm Linda	●			●					◕

LEGEND
❶ Open Source Space Computing System
❷ Commercial Space Computing System

● Implementation and Source Code available
◔ ◑ ◕ No Implementation but sufficient, good or extensive Information available
○ No Implementation and no Information available

Figure 4.6: Summary Classification of Space Based Computing Systems by Family K-S

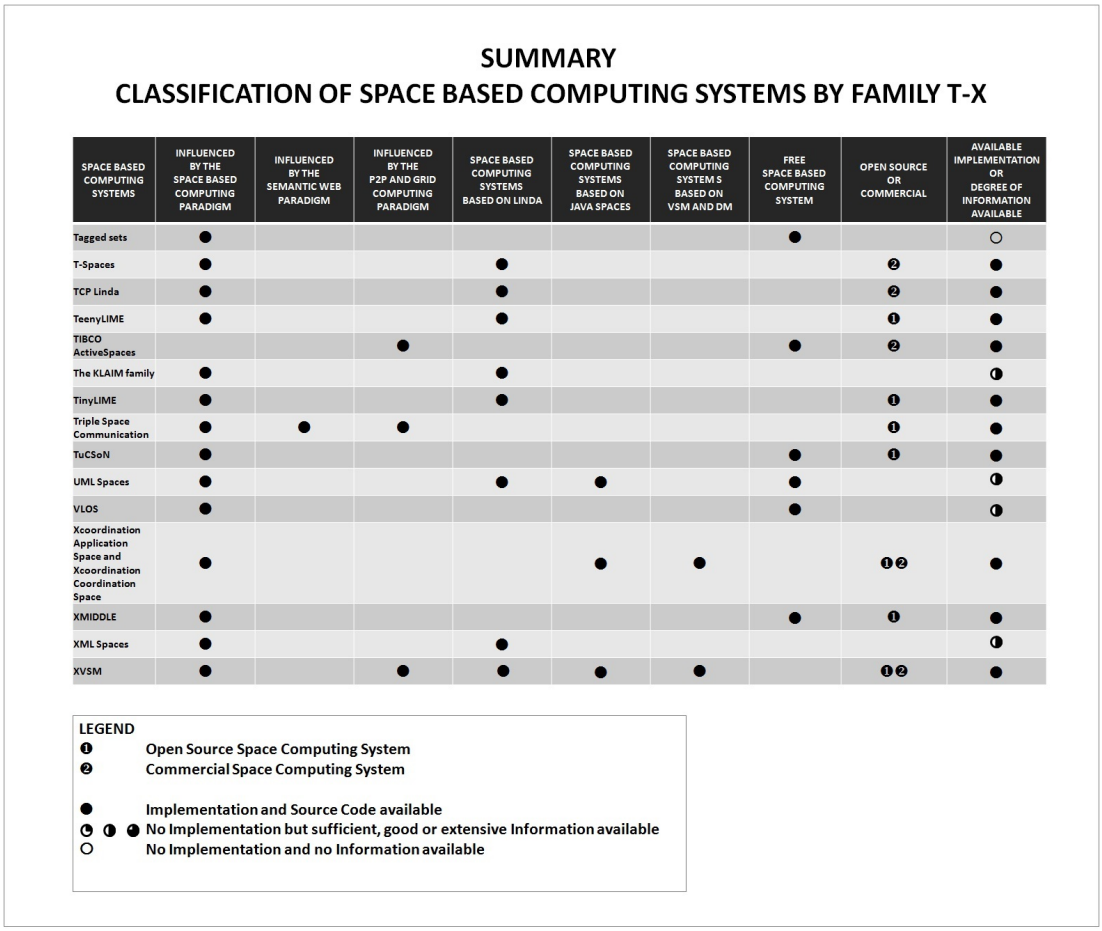


Figure 4.7: Summary Classification of Space Based Computing Systems by Family T-X

4.3 Classification by operations

In [Car89] it is already explained why Linda is a simple, powerful and quite elegant model. This is also due to the fact that Linda consists of only a few simple operations that combined with any programming language add up to a strong parallel programming dialect.
In this section I will present the most basic operations but also interesting extensions in this area. Furthermore I will talk about operations in the context of space based computing families. Further sections will discuss the topic of transactions and notifications in greater detail.
The last part of this section will provide a summary and an overview of the operations available per space based computing implementation.

4.3.1 Basic operations

Since Linda is the origin of all space based computing implementations it is no wonder that also the basic operations derive from Linda.

As written in an earlier chapter Linda provides four basic and two additional operations for accessing the tuple space. During the classification process it has become clear that three of these six operations proposed by Linda can be actually seen as the basic operations which are taken on by most of the space based computing implementations analysed in this thesis. These three operations are:

- `out (t)`: adds a tuple `t` to the tuple space and terminates. (non-blocking)
- `in (t)`: finds and removes a matching tuple`t` from the tuple space and terminates. (blocking)
- `rd (t)`: finds and reads a tuple `t` in the tuple space and terminates. (blocking)

The terms `out (t)`, `in (t)` and `rd (t)` are often used by space based computing systems which are directly connected to Linda or at least strongly influenced by it. However, there is another commonly used terminology which originates from Java Spaces.

- `write`: writes a tuple `t` to the tuple space and terminates. (non-blocking)
- `take`: finds and takes a matching tuple `t` from the tuple space and terminates. (blocking)
- `read`: finds and reads a tuple `t` in the tuple space and terminates. (blocking)

The traditional `in (t)` and `rd (t)` operations are blocking operations, which means that if an `in (t)` or `rd (t)` operation is executed, the operation blocks until it can be fulfilled [Sch08a]. Imagine the following example: An agent wants to carry out an `in (t)` operation but currently there are no entries in the space that are matching its search criteria. As a result the operation blocks until a matching tuple enters the tuple space and the operation can execute. One can already see that this behaviour is not always ideal and can cause a lot of blocked operations which at worst case, never unblock. Later we will see that there are operations that adress this issue.
After having discussed the three basic operations, the figures 4.8 and 4.9 show the space based computing implementations which are supporting these three basic operations. 4.8 shows all the space based computing implementations stongly connected to or at least influenced by the Linda family wheras figure 4.9 shows space computing systems following other families.
A great majority of the space based computing systems support these three basic operations, nevertheless there are a few spaces which do not support them at all, only partly or have certain specialities. Subsequently I will present these exceptions in a little more detail:

- **XVSM** supports all three basic operations but can be seen as a speciality since its operations work on container-level and not for the entire space, plus it offers the possibility of extensions. [Sch08a]
- **LACIOS** [42] [ZBS09] extends Linda and is a data-oriented coordination language which focuses on the design and implementation of multi agent systems used for transportation

[42] Language for Agent Contextual Interaction in Open Systems

applications. LACIOS disposes of four operations (`spawn`, `add`, `update` and `look`). The `add` operation works similar to the `out (t)` operation and adds a tuple to the tuple space. The `look` operation however, *enables agents for both the perception and retrieval of objects*[43] and therfore represents a compination of the `in (t)` and `rd (t)` operations proposed by Linda.

- **Linearizable Byzantine Tuple Space** [44] [NBCdSFCL07], [Bes06] offers a solution for ad hoc networks and mobile agents where simple operations use simple quorum-based protocols and more complicated operations use consensus-based protocols. The distinctive feature of the Linearizable Byzantine Tuple Space spaces lies in the fact that it only supports non-blocking `in (t)` and `rd (t)` operations.

- **Corso**[45] [Neu03] extends Linda through a virtual shared memory space of Java objects and has been presented in 2001 by the tecco Software Entwicklung AG, a spin-off of the Technical University of Vienna at the institute for computer languages. One can `create` and `destroy` objects. At creation time one has to decide if an object is of constant or variable nature. A constant object has only one fixed value whereas the value of a variable object can be overwritten. Most importantly CORSO allows the `writing` and `reading` of objects within one transaction. Objects can also be replicated and allow an OID.

- **InfoSpaces** [BMSV] addresses the problem of the interaction between ubiquitous devices by using tuple spaces at user level. The following simple drag-and-drop operations are supported: `copy out`, `move out`, `copy in` and `move in`. The main idea is to `move` and `copy` information between ubiquitious devices. Every device can decide which information is private and which information can be shared. The main difference to the basic operations proposed by Linda is that the `out (t)` operation is split into two operations, namely `move out` and `copy out`.

[43] [ZBH09] page 3.
[44] LBTS
[45] Coordinated Shared Objects

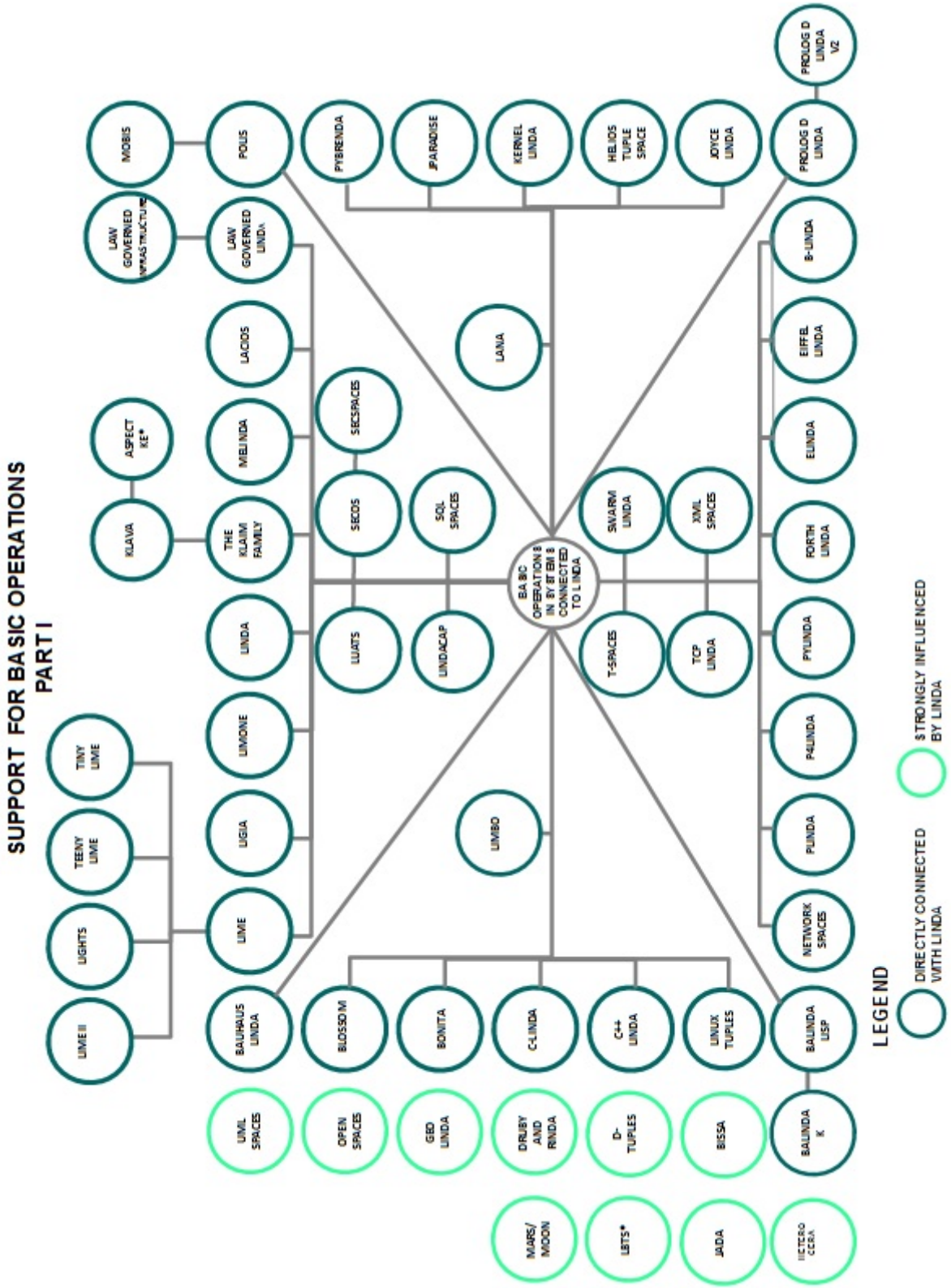


Figure 4.8: Support for Basic Operations - Part I

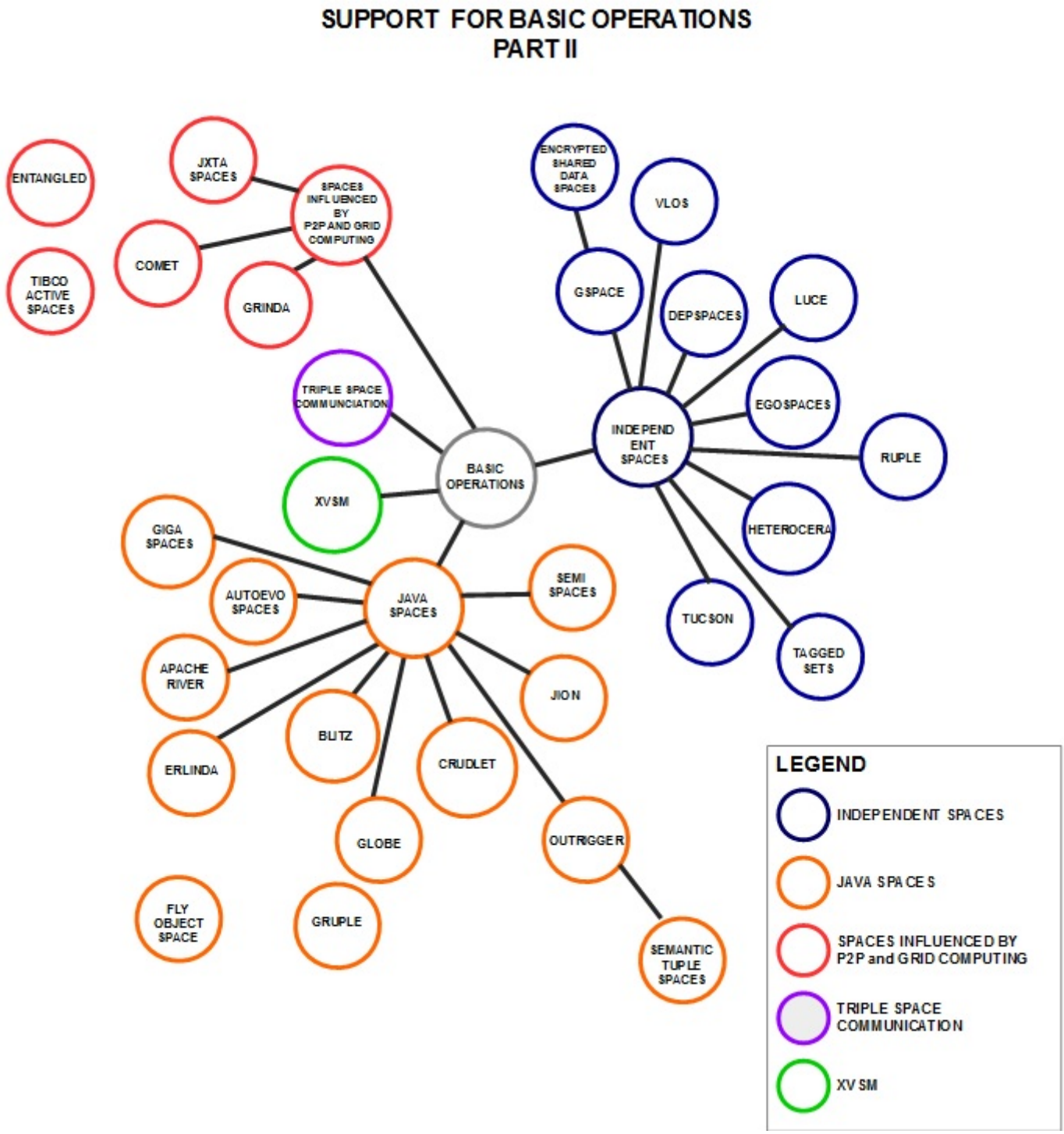


Figure 4.9: Support for Basic Operations - Part II

As written before the `in (t)` and `rd (t)` operations are blocking operations. Sometimes it

might be of interest that instead of blocking the operation a value is returned to the agent that no matching tuple could be found. Figure 4.10 shows all the space based computing implementations that support non-blocking operations. One can see that about half of all space based computing implementations support non-blocking operations.

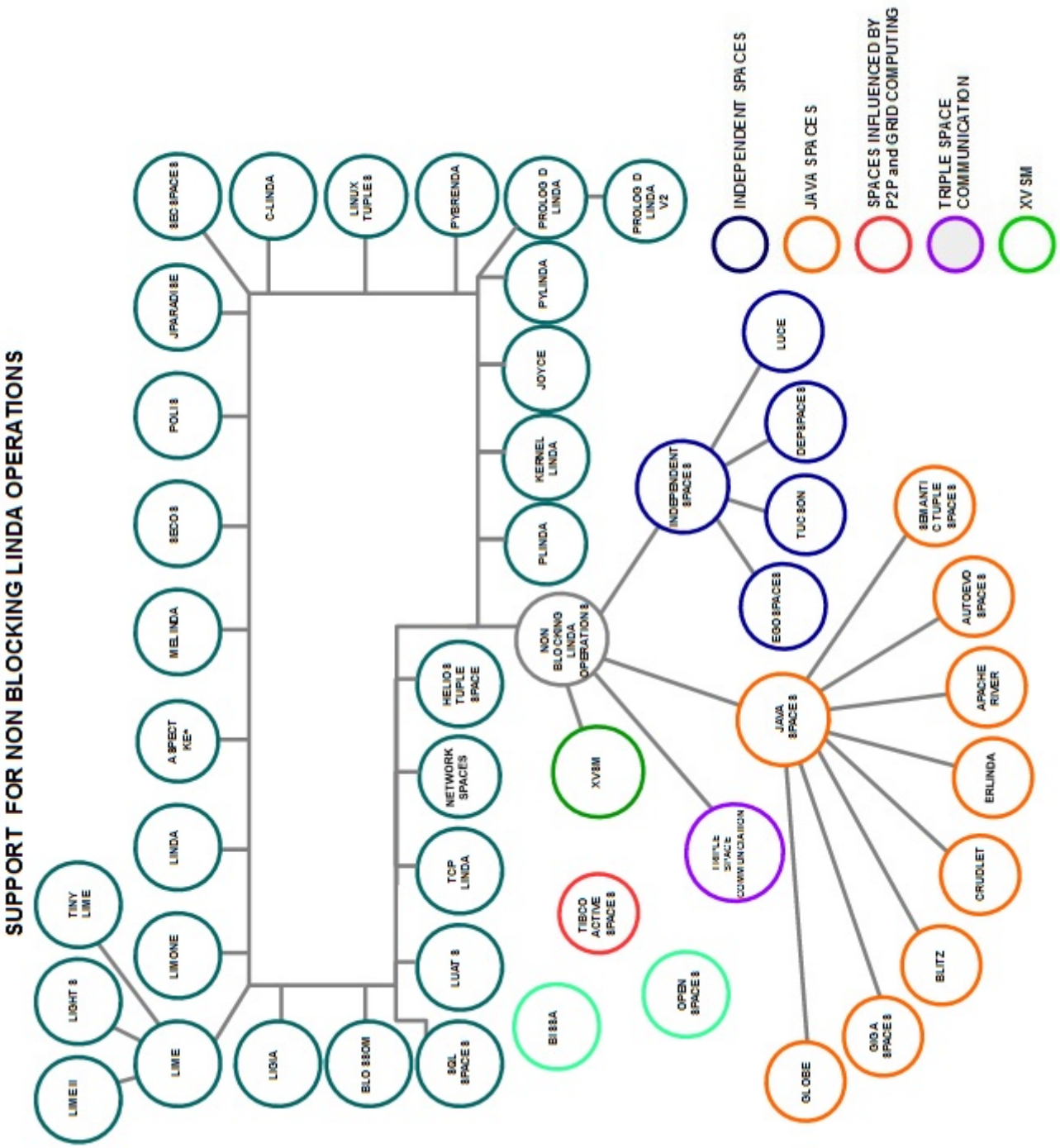


Figure 4.10: Support for Non Blocking Linda Operations

Like the `out (t)` operation the `eval (t)` operation can be seen as an `output` operation

because it places a tuple in the tuple space. Although the `eval (t)` operation is very similar to the operation `out (t)` it is different due to the fact that `eval (t)` creates an active tuple in the tuple space. This can be illustrated at the following basic example:
`eval (X (), Y ())` creates two processes `X ()` and `Y ()` *which are placed in the tuple space and are evaluated concurrently.*[46] This means that the tuple is unavailable for matching as long as the two processes `X ()` and `Y ()` are being evaluated. As soon as the evaluation of the two processes has finished, and a result is returned for example two integer values `1` and `2`, the active tuple turns into the passive tuple `(1, 2)`. `eval` then creates new processes and this is how parallelism is created in Linda and from where the term generative communication is derived. Figure 4.11 shows all space based computings systems. Here it can be seen that the `eval` operation is nearly exclusively used by systems that are strongly connected to Linda. Only Java Spaces and Erlinda also support the `eval` operation.

[46] [EZCdrei92] page 48.

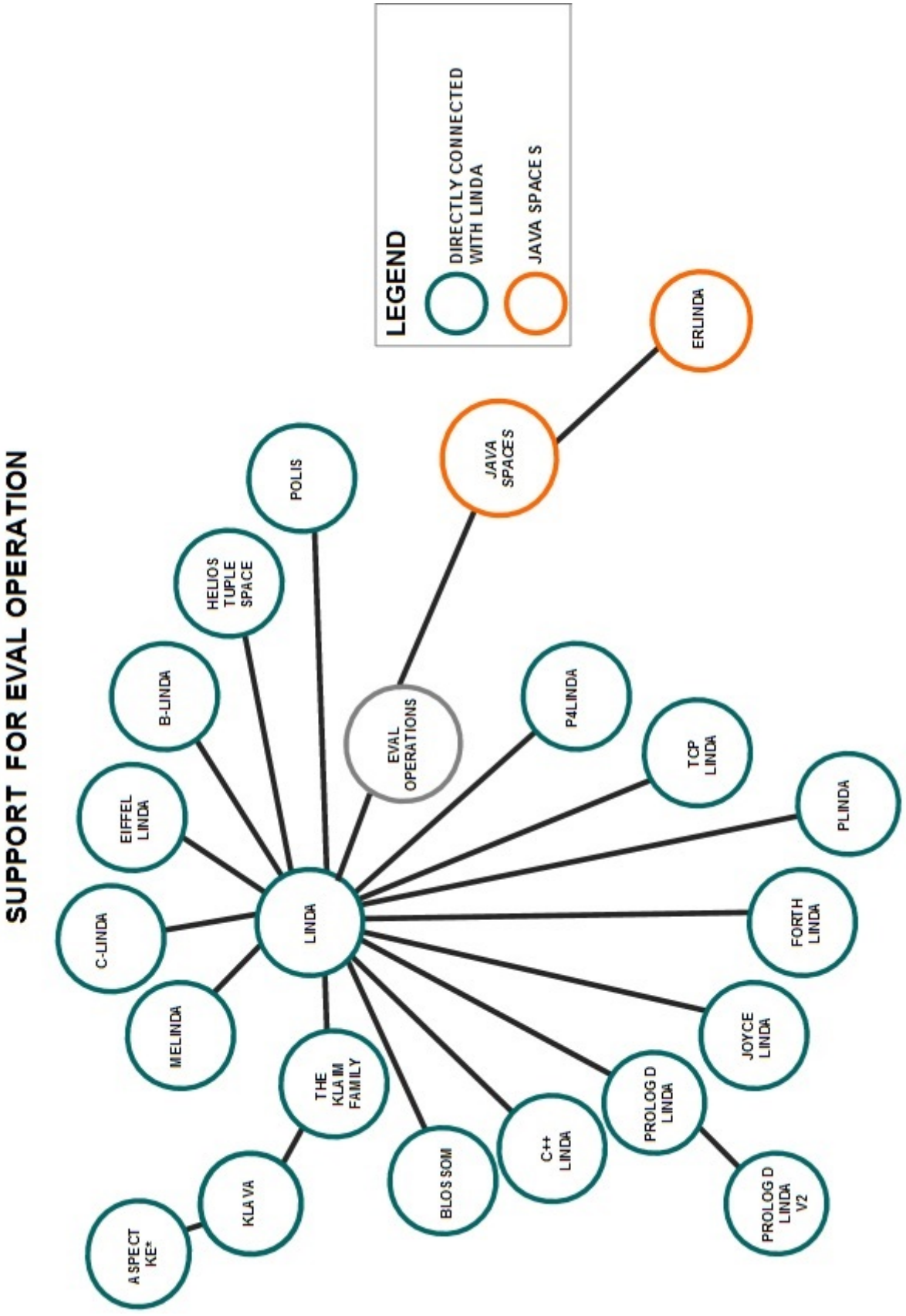


Figure 4.11: Support for The Eval Operation

4.3.2 Extended operations

In section 4.3.1 the basic operations were introduced but of course there are other interesting operations which are provided by spaces based computing systems in order to support certain application scenarios. Bulk operations are widely-used amongst space based computing systems, e.g Bonita [Row97], Giga Spaces[47], Grinda [CM08b], JXTA Spaces [Li01], Ruple [KLF04] or LIME[48] . These bulk-operations can range from `writing`, `reading` or `taking` multiple tuples to respectively from the tuple space [Sch08a]. Since a greater number of operations also means more possibilities T Spaces introduced an `add handler()` operator which allows developers to introduce new operators according to their needs. [49] The Helios Tuple Space library [CM08b] enhances the Linda model through additional library calls in the C Language, e.g. program termination, jobs can be split in portions and be given to free workers or mutual exclusion. Other implementations offer `cache`, `create`, `destroy` or `update` functionality, e.g. JavaSpaces, GigaSpaces, Helios Tuple Spaces or XVSM. Also `copy` and `move` operators can be found often, e.g. in Blossom or PyBrenda. Figure 4.12 shows which space based computing implementations offer extended primitives.

[47] `http://www.gigaspaces.com/`
[48] `http://lime.sourceforge.net/api/lime/LimeTupleSpace.html`
[49] `http://almaden.ibm.com/cs/TSpaces/papers/Cluster.pdf` page 5.

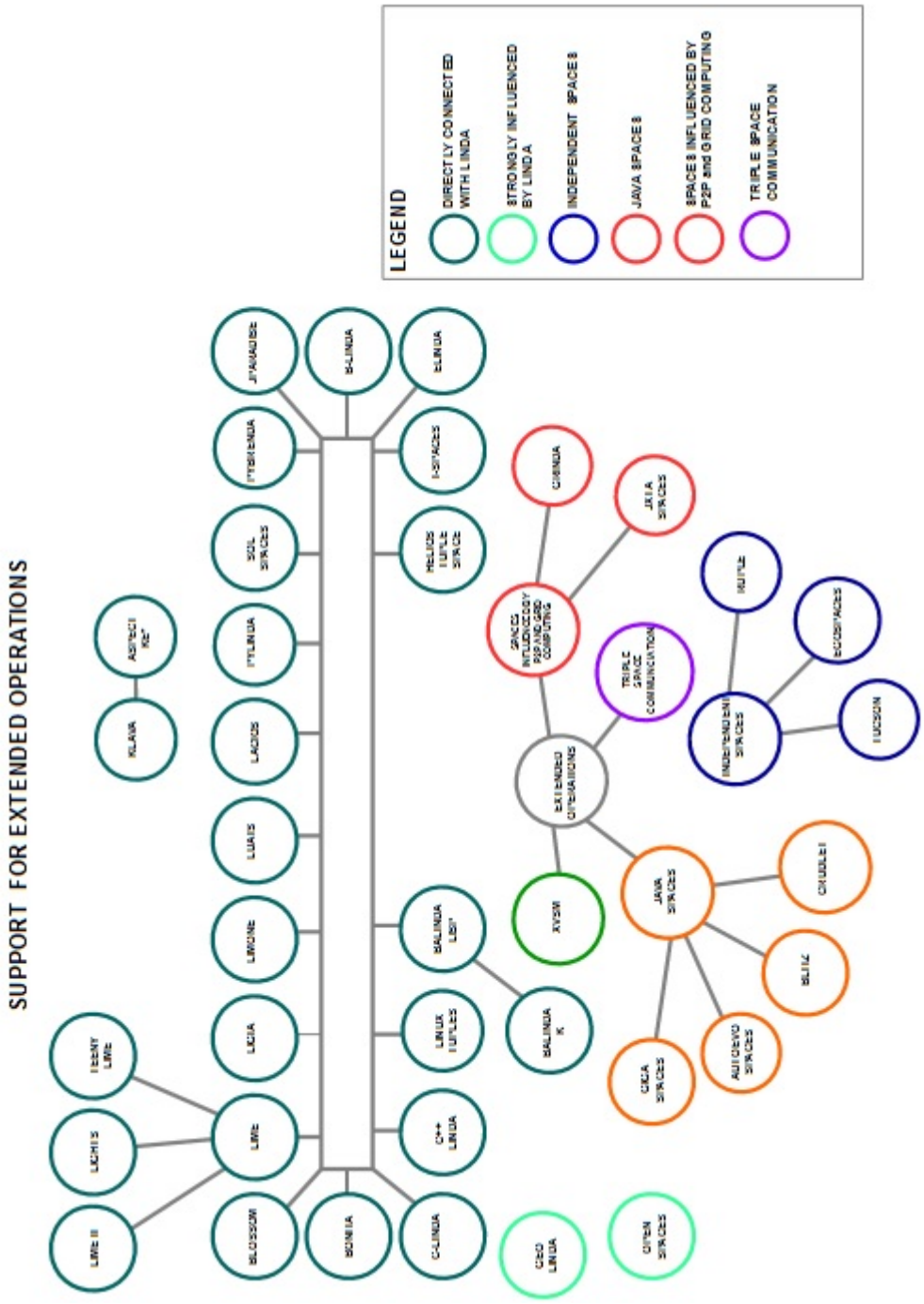


Figure 4.12: Support for Extended Operations

Figure 4.13 shows which extensions are supported by these space based computing implementations.

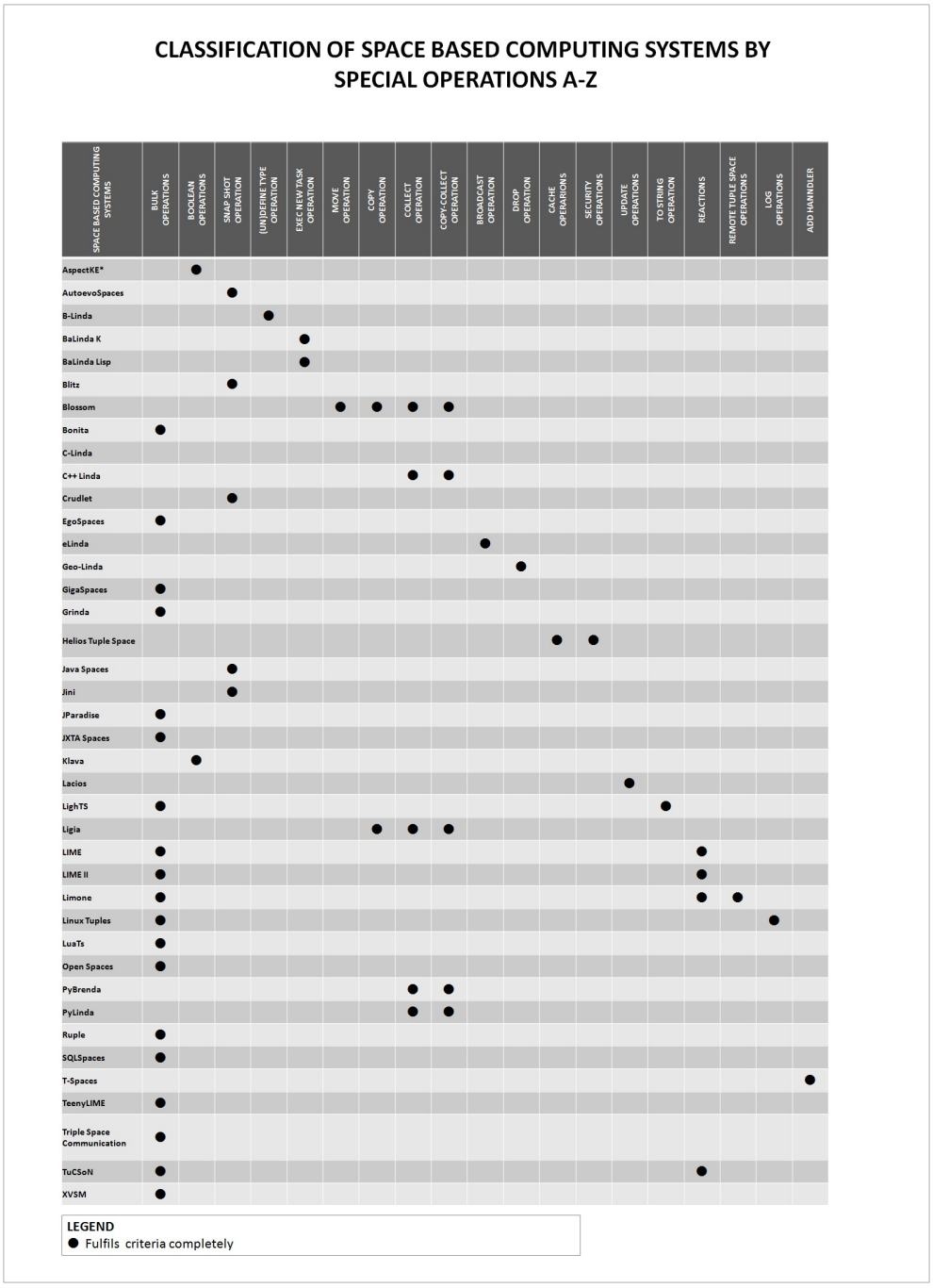


Figure 4.13: Extended Operations in Detail

4.3.3 Notifications

Having certain information without waiting all the time is without a doubt a very attractive feature. Application scenarios focusing on information monitoring or collaborative computing certainly rely on fast event notifications. Nearly half of all the space based computing implementations provide notification mechanisms, e.g. XVSM, UML Spaces [AR01], JavaSpaces, T Spaces, Apache River and Corso amongst others.
Members of the JavaSpaces family like Blitz, GigaSpaces, Apache River and JavaSpaces itself are working in general with event handlers. An agent can register to a certain event which is normally represented by a template. The operation `notify` *is used to register interest in the arrival of an entry into the space that matches a specified template.*[50] As soon as an entry has arrived that matches the template the caller gets notified by the space. In order to make this possible Jini introduces three entities: the event source, the remote event object and the remote event listener. In JavaSpaces the space acts as the event source that fires events *when entries are written into it and notifies processes that have registered interest in entries that match specified templates.*[51] This process is described in detail in the *Jini Distributed Event Specification*[52]. As soon as the event has occured and the process has been notified, the operation `notify` acts like the operations `read` or `take` in the sense that it either leaves the entry in the tuple space or takes it from the tuple space.
Corso also offers the possibility of notifications but does not work with an event handler. Therefore it can be seen as less suitable when it comes to scenarios where notifications are of importance.
Also XVSM offers notification services, which can listen on various operations in one container and be called back if such an operation is performed. Allowed values are `write`,`shift`, `read`, `take` and `destroy`.
In the summary section an overview of all the space based computing implementations offering notification services will be provided.

4.3.4 Transactions

The last sections have shown the variaty of operations which are available for space based computing systems. It is also obvious that these operations need to be structured in some way since the complexity of programmes and application scenarios is increasing steadily. Transactions offer an option to control such a complexity [JV04].
A transaction normally is composed of several operations which need all to be executed sucessfully so that they actually have an effect on the space. If one operation cann't be terminated sucessfully - the entire transaction is rolled back and the space remains unchanged. This behaviour fullfills the ideas of ACID properties[53]:

- *Atomicity: Each transaction runs to completion or has no effect at all.*

[50] http://www2.sys-con.com/itsg/virtualcd/java/archives/0509/mahapatra/index.html
[51] http://www2.sys-con.com/itsg/virtualcd/java/archives/0509/mahapatra/index.html
[52] http://river.apache.org/doc/specs/html/event-spec.html
[53] `http://itu.dk/people/pagh/idb11/KBL13.pdf`

- *Consistency: After a transaction completes, the integrity constraints are satisfied.*
- *Isolation: Transactions executed in parallel have the same effect as if they were executed sequentially.*
- *Durability: The effect of a committed transaction remains in the database even if the computer crashes.*

Spaces like Java-Spaces, TSpaces and many others try to integrate the ideas of space based computing with transactions. One can distinguish between two types of transactions: pessimistic transactions and optimistic transactions[54].
A pessimistic transaction locks all operations that are part of the transaction. This assures *that no other transactions can access these resources until the transaction is either committed or rolled back.*[55] Pessimistic transactions can lead to deadlocks. To handle this problem transactions are normally rolled back after a certain time if they do not have completed sucessfully after a certain time. Examples for pessimistic transactions are Blitz, BISSA, Erlinda, JION, Joyce Linda, GigaSpaces, P-Linda, SQL-Spaces and XVSM.
Optimistic transactions do not lock the operations involved in the transaction but check after the commit if all the operations were really available and not taken by another transaction. If an operation was not available an exception will be thrown. One may argue that optimistic transactions are more error-prone than pessimistic transactions when under pressure but the *better performance, scalability, and lower risk of hanging due to deadlock*[56] are clear advantages.
Corso for example uses optimistic transactions and TIBCO Active Spaces allows the use of both optimistic and pessimistic transactions.
LIME for example supports distributed transactions which are allowing transactions over the entire network. Also Blitz and Corso amongst others support distributed transactions [Sch08a].
In the summary all the space based computing systems supporting transactions are shown separately.

4.3.5 Summary

The figures 4.14 to 4.16 summarize all the operation types used by the different spaces and give an overview about all the spaces which support notifications and transactions. Forty of all space based computing implementations support transactions and 43 notifications. This shows although space based comuting systems aim to be simple and straight forward more and more complex application scenarios make it necessary to add additional features.

[54] `http://openjpa.apache.org/builds/1.2.3/apache-openjpa/docs/jpa_overview_trans.html`
[55] [Sch08a] page 14.
[56] `http://openjpa.apache.org/builds/1.2.3/apache-openjpa/docs/jpa_overview_trans.html`

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY OPERATIONS, NOTIFICATIONS AND TRANSACTIONS A-J

	OPERATIONS					NOTIFICATIONS	TRANSACTIONS
SPACE BASED COMPUTING SYSTEMS	BASIC LINDA OPERATIONS	EVAL OPERATION	NON BLOCKING LINDA OPERATIONS	EXTENDED OPERATIONS	SPECIAL CASE	NOTIFICATIONS	TRANSACTIONS
Apache River	●		●			●	●
AspectKE*	●	●	●	●			
AutoevoSpaces	●		●	●			
B-Linda	●	●		●			
BaLinda K	●			●			
BaLinda Lisp	●			●			
Bauhaus Linda	●						
BISSA	●		●			●	●
Blitz	●		●	●		●	●
Blossom	●	●	●	●			
Bonita	●			●			
C-Linda	●	●	●	●			
C++ Linda	●	●		●			
Comet	●					●	
Corso					●	●	●
Crudlet	●		●	●		●	●
D-Tuples	●					●	●
DepSpaces	●		●				
dRuby and Rinda	●						
EgoSpaces	●		●	●		●	●
Eiffel Linda	●	●					
eLinda	●			●			
Encrypted Shared Data Spaces	●						
Entangled	●						
Erlinda	●	●	●			●	●
Fly Object Space	●					●	●
Forth-Linda	●	●					
Gaia							
Geo-Linda	●			●		●	
GigaSpaces	●		●	●		●	●
Glinda						●	●
Globe	●		●				
Grinda	●			●		●	●
Gruple	●					●	●
Gspace	●						
Helios Tuple Space	●	●	●	●			
Heterocera	●						
HTML Page Spaces							
Info Spaces					●		
Jada	●						
Java Spaces	●		●	●		●	●
Jedi					●	●	●
Jini	●		●	●		●	●
JION	●					●	●
Joyce Linda	●	●	●				
JParadise	●		●	●			
JXTA Spaces	●			●		●	●

LEGEND
● Fulfils criteria completely

○ Fulfils criteria partly

Figure 4.14: Summary Classification of Space Based Computing Systems by Operations A-J

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY OPERATIONS, NOTIFICATIONS AND TRANSACTIONS K-S

	OPERATIOSN					NOTIFACTIONS	TRANSACTIONS
SPACE BASED COMPUTING SYSTEMS	BASIC LINDA OPERATIONS	EVAL OPERATION	NON BLOCKING LINDA OPERATIONS	EXTENDED OPERATIONS	SPECIAL CASE	NOTIFICATIONS	TRANSACTIONS
Kernel Linda	●		●				
Klava	●	●		●		●	●
L²imbo	●						
Lacios	○			●		●	
Lana	●					●	
Law-Governed Infrasturcture	●						
Law-Governed Linda	●						
LighTS	●		●	●			
Ligia	●		●	●			
Limbo	●						
LIME	●		●	●		●	●
LIME II	●		●	●		●	●
Limone	●		●	●		●	
Linda	●	●	●				
Lindacap	●						
Linearizable Byzantine Tuple Space	○					●	
LinqSpace	●		●			●	●
Linux Tuples	●		●	●			
LuaTs	●		●	●		●	●
LuCe	●		●			●	●
MARS/Moon	●					●	●
Melinda	●	●	●				
MobiS	●						
Network Spaces	●		●				
Open Spaces	●		●	●			
Open Wings							
P-Linda	●	●	●				●
P4 Linda	●	●					
PadSpace							
PoliS	●	●	●				
Prolog-D-Linda v2	●	●	●				
PyBrenda	●		○	●			
PyLinda	●		●	●			
Ruple	●			●			
Semantic Tuple Spaces	●		●			●	●
SecOS	●		●				
SecSpaces	●		●				●
SemiSpace	●					●	●
SmallSpaces							
SQLSpaces	●		●	●		●	●
Swarm Linda	●						

LEGEND
● Fulfils criteria completely

○ Fulfils criteria partly

Figure 4.15: Summary Classification of Space Based Computing Systems by Operations K-S

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY OPERATIONS, NOTIFICATIONS AND TRANSACTIONS T-X

	OPERATIONS					NOTIFICATIONS	TRANSACTIONS
SPACE BASED COMPUTING SYSTEMS	BASIC LINDA OPERATIONS	EVAL OPERATION	NON BLOCKING LINDA OPERATIONS	EXTENDED OPERATIONS	SPECIAL CASE	NOTIFICATIONS	TRANSACTIONS
Tagged sets	●						
T-Spaces	●			●		●	●
TCP Linda	●	●	○				
TeenyLIME	●			●		●	●
TIBCO ActiveSpaces	●		●			●	●
The KLAIM family	●	●					
TinyLIME	●		●			●	●
Triple Space Communication	●		●	●		●	●
TuCSoN	●		●	●			
UML Spaces	●					●	●
VLOS	●						●
Xcoordination Application Space and Xcoordination Coordination Space					●		
XMIDDLE					●	●	●
XML Spaces	●						
XVSM	●		●	●		●	●

LEGEND
● Fulfils criteria completely
○ Fulfils criteria partly

Figure 4.16: Summary Classification of Space Based Computing Systems by Family T-X

4.4 Classification by coordination concept

When talking about classification concepts for spaced based computing systems, the first questions which comes into mind is: Isn't Linda the main coordination concept for space based computing systems? This is quite true but the following section will show that there are application scenarios which are more efficient when using other coordination concepts like FIFO[57] or hashtables e.g.
I will start to introduce the specific coordination types, writing about their mode of operation as well as their advantages and disadvantages. In the summary section of this chapter I will classify the space based computing systems by the coordination concepts they support.

[57] First-In-First-Out

4.4.1 Linda coordination

Linda coordination was already discussed in detail in chapters 2.3.1 and 2.3.6 since it can be seen as the key coordination type for nearly all space based computing systems. Elements (tuples) are written to the space in a complete unordered way. To retrieve a suitable element a template is defined which is compared with all the elements in the space and only suitable matches are returned.
The advantage of this behaviour is that the space can handle multiple operations at the same time [Bar10].

4.4.2 First in first out

The FIFO[58] coordinator structures the elements within the space like a queue which makes it interesting for application scenarios where you want to read the first written element first. *Producer/consumer-like scenarios where data should be processed in the order it is written into the space*[59] can benefiit a lot from such an coordination type. Figure 4.17 illustrates how the FIFO concept works on a space.

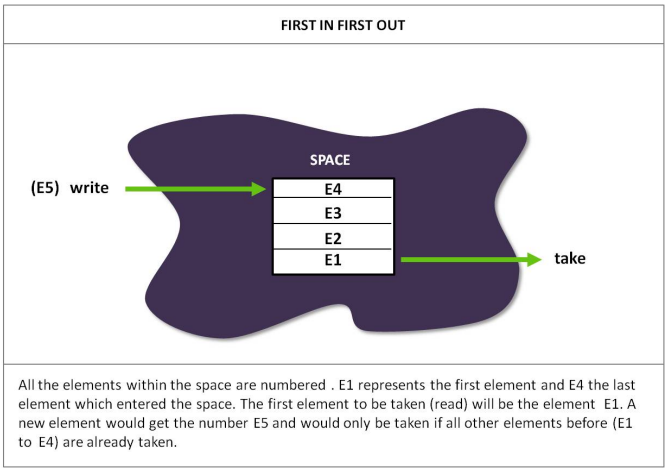


Figure 4.17: First In First Out

A possible disadvantage of using a FIFO coordinator can be seen in the fact that the right order of the elements has to be preserved under all circumstances [Bar10]. As a consequence if an element is blocked the operation is blocked and one might have to deal with long waiting times.

4.4.3 Last in first out

The LIFO[60] can be compared to a stack, which means that the element which entered the space last is taken out first. The advantages and disadvantages are quite similar to the FIFO coordina-

[58] first-in-first-out
[59] [Sch08a] page 9.
[60] Last-In-First-Out

tor. Figure 4.18 gives an overview over this coordination type.

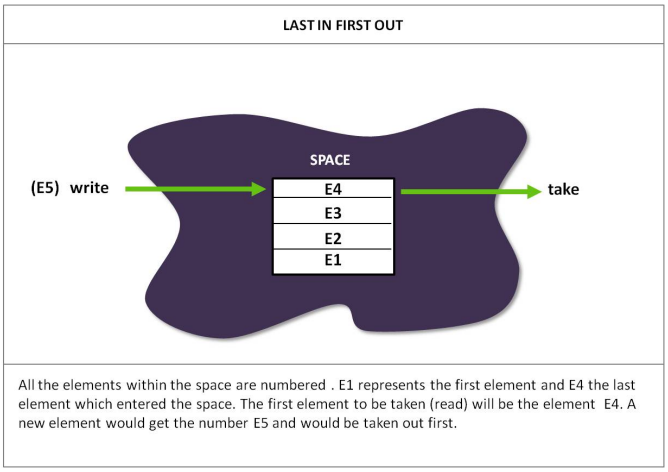


Figure 4.18: Last In First Out

4.4.4 Random

When using a random coordinator one does not want a predictable result because the random coordinator just sorts the elements in the space randomly before returning an element [Doe11]. [Doe11] also introduces the any operator which returns elements without a specific order without resorting them before. Figure 4.19 shows how the random coordinator works.

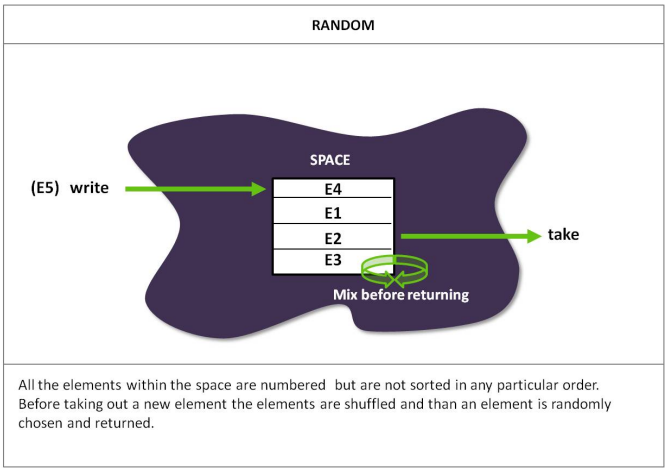


Figure 4.19: Random

4.4.5 Key Coordinator

The key coordinator is an explicit coordinator which assigns an unique key to an entry[61]. Figure 4.20 illustrates the behaviour of the key coordinator.

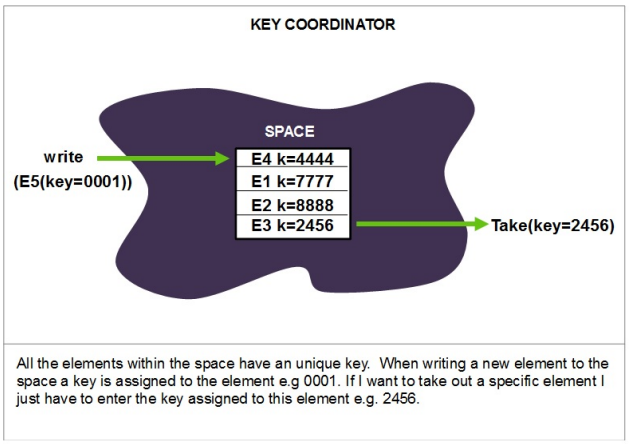


Figure 4.20: Key Coordinator

A good example for such a key are personnel Ids in a company e.g. When writing a new element to the space a key is automatically assigned to the element. As a result one can search elements quite easy because they all have an unique key one can refer to. A quite similar coordination type is the label coordinator which assigns a label to an element but not asking for uniqueness like the key coordinator [Bar10].

4.4.6 Coordination with hashtables

Adressing elements in an explicit way can also be achieved through hashtables[62]. Hashtables are not used very often by space based computing systems. This statement will be underlinded by the detailed classification in the summary section. [Sch08a] writes that this might be due to the fact that this coordination type is not so easily used with the Linda coordination. However when intelligently combined with space based computing system this solution can be quite powerful as for example shown in [BFK+11].

4.4.7 Summary

The last section has shown that there are several coordination types worth mentioning and considering in context with space based computing systems. A few coordination types like least recently used, FILO, or vector coordination amongst others have not been discribed in closer detail because they followed similar ideas like the cordination types presented. It is quite evident that most of the space based computing systems are supporting the Linda coordination

[61] `http://www.mozartspaces.org/2.2-SNAPSHOT/docs/MozartSpaces_Tutorial.pdf` page 18.

[62] `http://en.wikipedia.org/wiki/Hash_table`

type, namely 71 % of all space based computing systems. Apart from that one can say that other coordination types are not supported on a regular basis. This can be seen when looking at figure 4.21. The FIFO coordinator is the second frequently supported with thirteen percent followed by the key coordinator and hashtable coordinator with both six percent. Subsequently a detailed classification of all space based computing systems is shown in the figures 4.22 to 4.23:

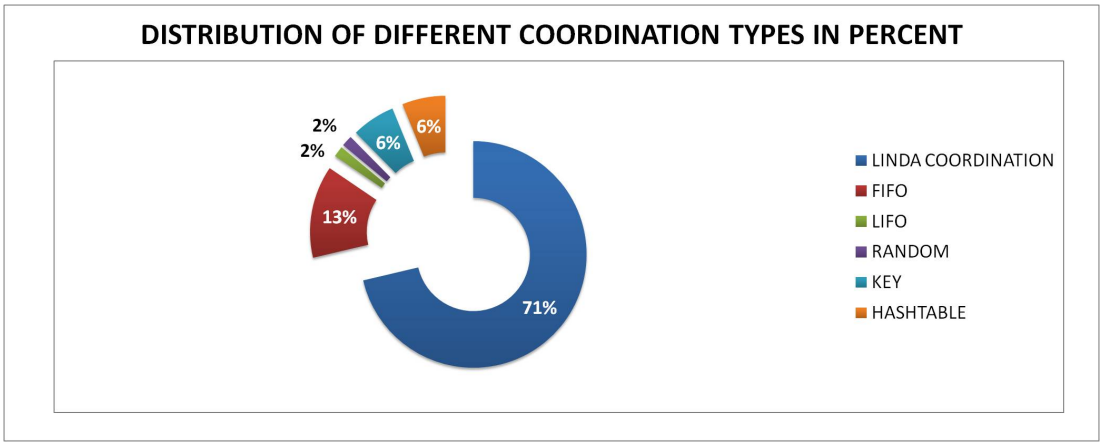


Figure 4.21: Distribution of Different Coordination Types in Percent

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY COORDINATION TYPES A-J

SPACE BASED COMPUTING SYSTEMS	LINDA COORDINATION	FIFO COORDINATION	LIFO COORDINATION	RANDOM COORDINATION	KEYS COORDINATION	HASHTABLE COORDINATION	OTHERS
Apache River	●	●					
AspectKE*	●						
AutoevoSpaces	●						
B-Linda	●						
BaLinda K	●						
BaLinda Lisp	●						
Bauhaus Linda	●						
BISSA	●						
Blitz	●	●					
Blossom		●					
Bonita	●						
C-Linda	●						
C++ Linda	●						
Comet	●						
Corso							●
Crudlet	●	●					
D-Tuples						●	
DepSpaces	●						
dRuby and Rinda	●						
EgoSpaces	●						
Eiffel Linda	●						
eLinda	●	●					
Encrypted Shared Data Spaces	●				●		
Entangled	●					●	
Erlinda	●	●					
Fly Object Space	●						
Forth-Linda	●						
Gaia							
Geo-Linda	●						
GigaSpaces	●	●			●		
Glinda	●						
Globe	●	●					
Grinda	●						
Gruple	●						
Gspace	●						
Helios Tuple Space	●						
Heterocera	●						
HTML Page Spaces							
Info Spaces							
Jada	●	●					
Java Spaces	●	●					
Jedi	●						
Jini	●	●					
JION	●						
Joyce Linda	●						
JParadise	●						
JXTA Spaces	●	●					

LEGEND
● Fulfils criteria completely

Figure 4.22: Summary Classification of Space Based Computing Systems by Coordination Types A-J

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY COORDINATION TYPES K-S

SPACE BASED COMPUTING SYSTEMS	LINDA COORDINATION	FIFO COORDINATION	LIFO COORDINATION	RANDOM COORDINATION	KEYS COORDINATION	HASHTABLE COORDINATION	OTHERS
Kernel Linda	●						
Klava	●					●	
L²imbo	●						
Lacios	●						
Lana	●						
Law-Governed Infrasturcture	●						
Law-Governed Linda	●						
LighTS	●						
Ligia	●						
Limbo	●						
LIME	●						
LIME II	●						
Limone	●					●	
Linda	●						
Lindacap	●				●		
Linearizable Byzantine Tuple Space	●	●					
LinqSpace							
Linux Tuples	●						
LuaTs	●						
LuCe	●						
MARS/Moon	●	●					
Melinda	●						
MobiS	●						
Network Spaces	●						
Open Spaces	●						
Open Wings	●						
P-Linda	●						
P4 Linda	●						
PadSpace							
PoliS	●						
Prolog-D-Linda v2	●						
PyBrenda	●						
PyLinda	●						
Ruple	●						
Semantic Tuple Spaces	●						
SecOS	●					●	
SecSpaces	●						
SemiSpace	●						
SmallSpaces							
SQLSpaces	●						
Swarm Linda	●						

LEGEND
● Fulfils criteria completely

Figure 4.23: Summary Classification of Space Based Computing Systems by Coordination Types K-S

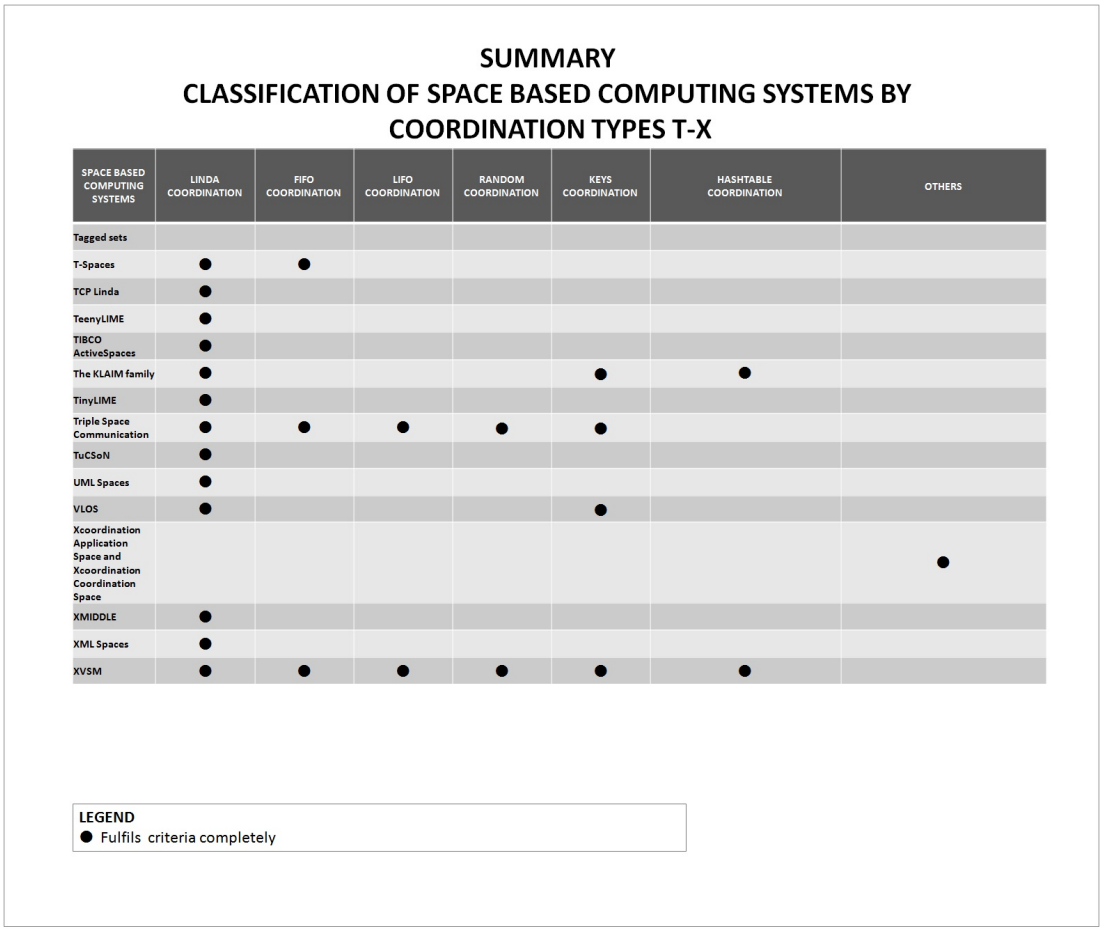


Figure 4.24: Summary Classification of Space Based Computing Systems by Coordination Types T-X

The summary shows that without a doubt XVSM is the most versatile space based computing system when it comes to coordination types. This is due to the fact that it works with containers which allow these different coordinators and therefore guarantee more freedom for the developer. It suppports all of the above mentioned coordination types.
Many space based computing systems influenced by JavaSpaces support a FIFO ordering for the entries in the space [Sch08a], e.g. Blitz[63], Apache River[64] or GigaSpaces[65] amongst others.
Hashtable or key coordinators are rarely used. *Entangled is a distributed hash table (DHT) based on Kademlia, as well as a peer-to-peer tuple space implementation.*[66] Also GigaSpaces offers the possibility to use the space like a has table, through defining a key scheller1.
Coordination types like LIFO or RANDOM are only supported by XVSM which leads to the conclusion that they are nice to have coordinators but not a must.

[63] `http://www.dancres.org/blitz/`
[64] `http://river.apache.org/`
[65] `http://www.gigaspaces.com/xap/overview`
[66] `http://entangled.sourceforge.net/`

When it comes to space based computing systems that focus on security aspects key coordinators are used more frequently, e.g. in VLOS [CM03] or SecOS [VBO03].
Corso once more follows a totally different approach. Objects in Corso have a unique identifier which can be addressed directly. The function is quite similar to a hashtable but a bit more straight forward [Ang03], [Sch08a].
Another space based computing system which might be interesting in this context is SwarmLinda [TM03]. Although following Linda coordination to a certain extent SwarmLinda also shows that organized behavior (here following the behavior of ants) can be implemented based on a few simple rules. Which leads to tuples ordered after there contence and after certain rules.

4.5 Classification by substructures

Deciding to classify space based computing system by substructures was not an obvious choice from the beginning. The reason for that is that on a first glance you might say that apart from a few space based computing systems not many use substructures but looking deeper into this subject, there are more space based computing systems that make use of substructures than one might think.
First of all lets define what is meant by substructure in context with this thesis. A substructure can be seen as a sub unit of a space which however is bigger than a single data element, e.g. a tuple [Sch08a]. So substructures could be spaces within the main space or we could think of an hierachical structure in a space where entities are ordered in some kind of way.
Concerning substructures XVSM is one of the few space computing system that uses real substructures, namely containers [Sch08a]. Containers can also refer to each other and are therfore capable to create hierachies. In contrast to XVSM, Corso does not work with containers but uses so called identifiers. These identifiers can be stored in various shared objects and so again hierachical dependencies can be created.
Another space based computing system which works with substructures is Ligia [MW98]. Ligia focuses mainly on the topic of garbage collection and also uses a structure mechanism to efficiently work. Ligia uses so called handles which represent nothing else than the names of the tuple space. This handles can be stored in another tuple spaces and so one is able to build a dependency structure between the different spaces. The dependencies are shown through directed arcs (called bridges) which are connecting the handles to graphes. This graphes facilitate the process of garbage collection because now one can see if one can really delete a tuple space without unknowingly destroying valid entities.
Also Lindacap [UDI09] uses references in order two build up a hierachy between the entities in order to realize a garbage collection mechanism.

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY SUBSTRUCTURES

SPACE BASED COMPUTING SYSTEMS	SUPPORTS SUBSTRUCTURES	SPACE BASED COMPUTING SYSTEMS	SUPPORTS SUBSTRUCTURES	SPACE BASED COMPUTING SYSTEMS	SUPPORTS SUBSTRUCTURES
Apache River	●	Helios Tuple Space		Network Spaces	
AspectKE*		Heterocera		Open Spaces	
AutoevoSpaces		HTML Page Spaces		Open Wings	●
B-Linda		Info Spaces		P-Linda	
BaLinda K		Jada	●	P4 Linda	
BaLinda Lisp		Java Spaces	●	PadSpace	
Bauhaus Linda		Jedi	●	PoliS	●●●
BISSA	●●	Jini	●	Prolog-D-Linda v2	
Blitz	●	JION		PyBrenda	
Blossom	●	Joyce Linda		PyLinda	
Bonita		JParadise		Ruple	
C-Linda		JXTA Spaces	●	Semantic Tuple Spaces	●●
C++ Linda		Kernel Linda		SecOS	
Comet	●●	Klava		SecSpaces	
Corso	●●●	L²imbo	●●	SemiSpace	
Crudlet	●	Lacios		SmallSpaces	
D-Tuples	●	Lana	●	SQLSpaces	
DepSpaces		Law-Governed Infrasturcture		Swarm Linda	●●
dRuby and Rinda		Law-Governed Linda		Tagged sets	
EgoSpaces		LighTS		T-Spaces	
Eiffel Linda		Ligia	●●	TCP Linda	
eLinda		Limbo	●●	TeenyLIME	●●
Encrypted Shared Data Spaces		LIME	●●	TIBCO ActiveSpaces	
Entangled		LIME II	●●	The KLAIM family	●
Erlinda		Limone	●●	TinyLIME	●●
Fly Object Space	●	Linda		Triple Space Communication	●●●
Forth-Linda		Lindacap		TuCSoN	●●
Gaia		Linearizable Byzantine Tuple Space		UML Spaces	
Geo-Linda	●	LinqSpace	●●●	VLOS	
GigaSpaces	●	Linux Tuples		Xcoordination Application Space and Xcoordination Coordination Space	●●●
Glinda	●	LuaTs		XMIDDLE	●●●
Globe		LuCe		XML Spaces	
Grinda	●	MARS/Moon	●	XVSM	●●●
Gruple	●	Melinda			
Gspace	●●	MobiS	●●●		

LEGEND
●●● Substructures and ordering mechanisms
●● Substructures (mainly subspaces)
● Ordering mechanisms

Figure 4.25: Summary Classification of Space Based Computing Systems by Substructures

The idea of using multiple tuples spaces has been adressed by various space based computing systems in order to address *issues of performance, partitioning and scalability*[67]. L^{2}imbo

[67] [DFBW98] page 5.

[DFBW98], Limbo [DWFB97], Limonelimone, GSpaces [RCVS04], BISSA [WWF+10] and Comet [LP05] amongst others. In this context also MobiS is very intersting since it works with multiple tuple spaces which are ordered in a tree structure. The tuple spaces are free to move and so their position within the tree can move. Also Xmiddle [MCZE02] focuses on the hierachical concept by using a tree structure to order data on mobile devices. The mobile devices can exchange data by addressing certain parts of this tree structure.
TinyLIME [CGG+05a] as well as the other members of its family(LIME [MPR01],TeenyLIME [CMP+09]) make use of tuple spaces within tuple spaces. The main space would be the federated tuple space which allows mobile devices in reach from each other to communicate. The agents on a mobile device also share one tuple space with each other and every host has its own private tuple space. This helps to keep a reasonable communication in a mobile environment alive.
Space based computing implementations like Blitz [Unk12], GigaSpaces [Pap05] or OpenWings [Bie12] are able to create an inheritance structure through using the subclasses provided by JavaSpaces [FAH99] [Sch08a]. This means simply spoken that searching for one class would also return available sublclasse if they are in the same space.
Figure 4.25 shows which spaces support substructures. Those that have only one point are basically focusing on ordering the elements within in the space and those which have two points really use some kind of substructure within the space, e.g. containers or other spaces. Those marked with three points support substructures and some kind of ordering mechanism, e.g. XVSM.

4.6 Classification by data type

Space based computing systems are nothing without the data they manage. Therefore it is also an interesting question which kind of data can be stored within in the space and in which structure the data needs to have. Although tuples are of course the main data type there are are also other data types. Subsequently I will present the following data types which are relevant in the context of space based computing systems:

- Primitive Types, e.g. character, integer, floating-point number, fixed-point number, string, boolean or references.
- Tuples
- Objects

[Sch08a] has stated in his paper that different space base computing systems have rather different approaches, which is quite true although one can say that the basic ideas behind the used data types are often quite similar.
Figure 4.26 shows which spaces support which data types. Each space based computing system is classified after the data types it supports. T stands for tuples, O stands for objects, P for primitives and S for special data types solutions.

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY DATA TYPES

SPACE BASED COMPUTING SYSTEMS	SUPPORTS DATA TYPES	SPACE BASED COMPUTING SYSTEMS	SUPPORTS DATA TYPES	SPACE BASED COMPUTING SYSTEMS	SUPPORTS DATA TYPES
Apache River	O	Helios Tuple Space	T	Network Spaces	T
AspectKE*	T	Heterocera	T	Open Spaces	T
AutoevoSpaces	O	HTML Page Spaces		Open Wings	O
B-Linda	T	Info Spaces		P-Linda	T
BaLinda K	T	Jada	O	P4 Linda	T
BaLinda Lisp	T	Java Spaces	O	PadSpace	
Bauhaus Linda	T	Jedi	T	PoliS	T
BISSA	T	Jini	O	Prolog-D-Linda v2	T
Blitz	O	JION	T	PyBrenda	T
Blossom	T	Joyce Linda	T	PyLinda	T
Bonita	T	JParadise	T	Ruple	O
C-Linda	T	JXTA Spaces	T	Semantic Tuple Spaces	S
C++ Linda	P, S	Kernel Linda	T	SecOS	T
Comet	P	Klava	O	SecSpaces	T
Corso	P	L²imbo	T	SemiSpace	
Crudlet	O	Lacios	T	SmallSpaces	
D-Tuples	T	Lana	T	SQLSpaces	T
DepSpaces	O	Law-Governed Infrasturcture	T, O	Swarm Linda	T
dRuby and Rinda	O	Law-Governed Linda	T, O	Tagged sets	T
EgoSpaces	T	LighTS	T, S	T-Spaces	O
Eiffel Linda	T	Ligia	T	TCP Linda	T
eLinda	T	Limbo	T	TeenyLIME	T
Encrypted Shared Data Spaces	T	LIME	T	TIBCO ActiveSpaces	O
Entangled	T	LIME II	T	The KLAIM family	T
Erlinda	O	Limone	T	TinyLIME	T
Fly Object Space	O	Linda	T	Triple Space Communication	T
Forth-Linda	T	Lindacap	T	TuCSoN	T
Gaia		Linearizable Byzantine Tuple Space	T	UML Spaces	T
Geo-Linda	T	LinqSpace	S	VLOS	T
GigaSpaces	S	Linux Tuples	T	Xcoordination Application Space and Xcoordination Coordination Space	S
Glinda	P, T	LuaTs	T	XMIDDLE	O
Globe	T	LuCe	T	XML Spaces	O
Grinda	O	MARS/Moon	O	XVSM	S
Gruple	O	Melinda	T		
Gspace	S	MobiS	T		

LEGEND
O Objects
T Tuples
P Primitives
S Specialities

Figure 4.26: Summary Classification of Space Based Computing Systems by Data Types

In most of the space based computing systems data is stored in tuples. A tuple is an ordered set of values. The separator for each value is often a comma (depending on the rules of the particular language). Common uses for the tuple as a data type are:

- *Passing a string of parameters from one program to another:*
- *Representing a set of value attributes in a relational database.*[68]

Since tuples can contain a mixture of other data types they can be used in different shapes. That might also be the reason why over 70 % of the analyzed space based computing systems use tuples as data types. Of course Linda makes use of tuples as data types. Many direct Linda descendants, like Melinda [Hup90], Pylinda [Wil12], EiffelLinda [Jel90] or Bauhaus Linda [CGZ95] amongst others, make use of tuples.
A space based computing system that uses primitives as data types instead of tuples is Corso [JV04]. The primitives supported are: integer, character, string, raw, OID, structure and stream.
JavaSpaces uses so called entries as data types. The Jini Entry Specification defines entries as follows: *Entries are designed to be used in distributed algorithms for which exact-match lookup semantics are useful. An entry is a typed set of objects, each of which may be tested for exact match with a template.*[69]. Jada [CR97] also uses objects as data types reffered to as items. Other space based computing systems using objects are Blitz[70], Erlinda[71] and Openwings[72].
Other space based computings systems allow more flexibility. XVSM for example is not constrained to specific data types. *Restrictions are only given when using tuple matching, or when interoperability is needed.*[73] CPP LINDA allows to develop own data types apart from supporting the common ones [DF96b]. GigaSpaces uses annotations which make it possible to decide which fields are entered into the data space. Also interoperabilty between programming language is possible because of the annotations.

4.7 Classification by extensibility

If a space based computing system supports extensibility or not cannot be answered easiliy and the truth is, in some cases the answer to this question lies more in the eyes of the beholder than anything else. Therefore it is wise to start with a definition of extensibilty in the context of space based computing systems. This definition is meant to help during the qualification process but does not try to be exhaustive since this would go beyond the scope of this thesis.
Subsequently I will write in more detail about the space based computing systems which support extensibility in one or the other way. Figure 4.27 shows which space based computing systems are taken into consideration.

[68] `http://searchcio-midmarket.techtarget.com/definition/tuple`
[69] `http://river.apache.org/doc/specs/html/entry-spec.html`
[70] `http://www.dancres.org/blitz/`
[71] `code.google.com/p/erlinda/`
[72] `http://www.openwings.org/`
[73] [Sch08a] page 12.

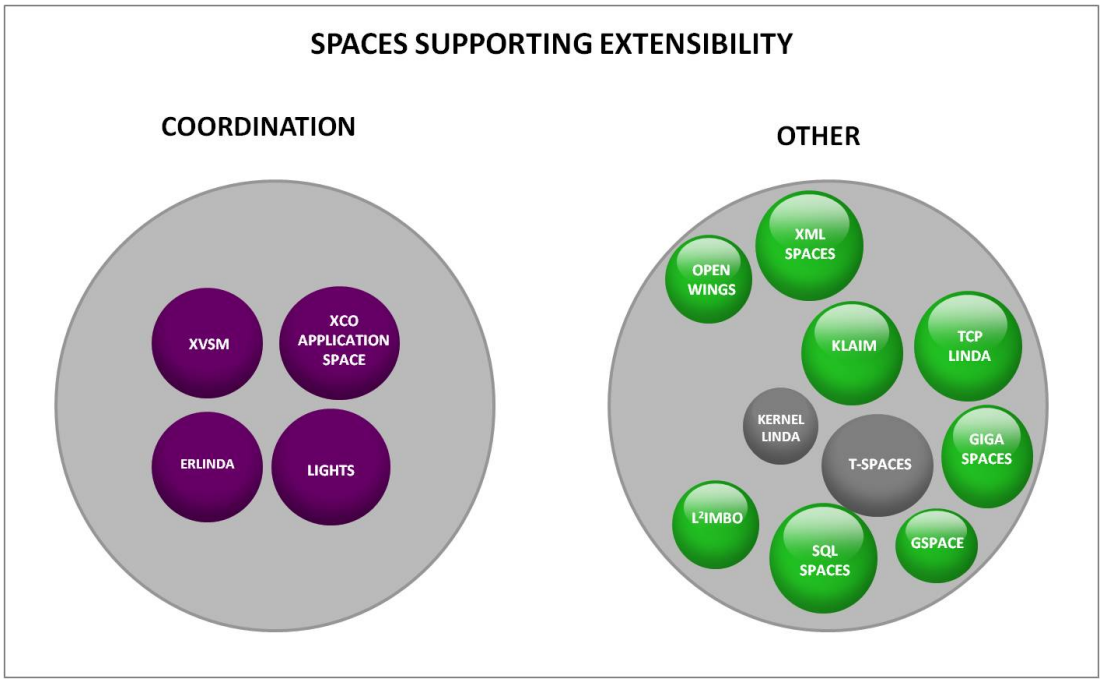


Figure 4.27: Spaces Supporting Extensibility

What can be seen at a glance is that one hardly can classify extensibility into equally distributed groups. Every system classified focuses on a different topic which is strongly influenced by the application scenario of the patricular space based computing system.
Four space based computing systems focus on offering extensibility to coordinational topics. The pioneer in this area is XVSM which not for nothing stands for eXtensible Virtual Shared Memory. Chapter 2.3.6 already introduced XVSM and talked in detail about the containers used within the tuple space. These containers provide all sorts of different coordination types and therfore every container can behave exactly like the developer wants the entries in the container to behave. Another space based computing system which addresses this topic is Erlinda[74]. Erlinda offers different ways to deal with coordination, e.g. the master slave paradigm, message queues or the classical Linda model amongst others. Having said this, Erlinda is not as flexible as the other space base computing systems in this category. Also the Xcoordination Application Space[75] which belongs to the XVSM family offers the possibility of costumizable communication channels. The forth space based computing system is called LighTS [BCP05] and has a framework which allows to extend the tuple space in many ways, *including changing the back-end implementation, redefining the matching semantics, and providing new constructs.*[76] Programmers can implement their own tuple classes since the space itself just works with the ITuple interface [Sch08a].

[74] `http://code.google.com/p/erlinda/`
[75] `http://www.xcoordination.org/application_space`
[76] [BCP05] page 1.

There are also other ways to extend a space based computing system. Subsequently all the other space based computing systems presented in figure 4.27 are presented shortly and it is explained why one could argue that this systems support extensibility.
GSpace [RCVS04] is a distributed shared data space that supports a variety of distribution policies. And in this sense it is also extensible since it supports different replication policies the developer can choose from.
Openwings[77] is a set of open system specifications for a framework that enables the development of highly available, secure, distributed systems for mission critical applications where systems are likely to come and go in an ad-hoc fashion. Openwings is an abstraction on top of various service discovery mechanisms, including Jini and provides a component framework for Service-Oriented Programming[78]:

- *Container services addresses the need for process lifecycle management, clustering/load balancing, security, handling of mobile code and overall service availability in a distributed environment.*[79]
- *Component services provide the mechanism to publish services as well as discover and use services that have been created and published by others.*[80]
- Connector services support component to component communication.
- Install services
- Context services
- Management services
- Security services

All this services can be used or not which leads to the conclusion that Openwings supports extensibilty. Also when using **XVSM** the developer can configure its own system by adding or unadding pluggable components (lifecycle management, monitoring, encryption etc.).
The L^{2}imbo, as presented in [DFWB98a] and in [DFWB98b] extends the classical Linda model. One of the extensions made to L^{2}imbo are the so called system agents which offer a variation of different services. Since this services can be choosen from, one might this also call extensibility.
GigaSpaces scales very well and focuses on its extensibilty in general. It tries to enable the partitioning of data into self-contained processing units which nearly can handle any load and dynamically allocate resources for optimized utilization.[81].

[77] http://www.openwings.org
[78] SOP
[79] http://www.openwings.org/openwings-11/tutorial/Trail_Introduction/04_Container_Services.html
[80] http://www.openwings.org/openwings-11/tutorial/Trail_Introduction/03_Component_Services.html
[81] http://www.gigaspaces.com/xap-high-availability-load-balancing/how-it-works

TCP's Linda architecture is modular and its semantics are the same as in Linda and an content-addressable tuple space, + makes it easier to build applications. One could argue that a modular architecture means that extensibilty is supported to a certain degree.
KLAIM[82] is a space-based computing implementation *designed to program distributed systems consisting of several mobile components that interact through multiple distributed tuple spaces.*[83] The development of KLAIM has been mainly influenced by process calculus (π-calculus) and the Linda coordination model with the aim to improve issues like scalability and modularity. Over the time various KLAIM implementations and extensions to it have been presented. Since there are so many extensions to Klaim itself which can be all intermingled with each other one could say that is extensibilty too.
XMLSpaces.NET implements the Linda concept as a middleware for XML documents on the .NET platform and it *is extensible in that it supports a hierarchy of matching relations on tuples and an open set of matching amongst data, documents and objects.*[84]
SQL[85]**Spaces** combine the idea of the tuple spaces concept with a relational database that can be seen as the backbone of the implementation but can hardly be noticed by the developer because of the features, e.g. notifications, expiration, versioning and extended query mechanisms, which are offered by the tuple spaces. The different query mechanisms could been described as extensibility but this is not straight forward.
Last but not least we close this section with XVSM and the topic of aspects. An Aspect in XVSM allows to implement an extension to the existing XVSM functionality[86]. Many features can be implemented using aspects, e.g. logging or notifications amongst others. The basic trick is to integrate code before and after each space operation which is executed. The code is executed automatically and extend the functionality of any operation executed. This of course is exetensibility.
Subsequently an overview of all space based computing systems is given in the context of extensibility. Figure 4.28.

[82] Kernel Language for Agents Interaction and Mobility
[83] [BBDN+03] page 88.
[84] [TLN04] page 1.
[85] Structured Query Language
[86] `http://www.mozartspaces.org/2.0/docs/MozartSpaces_Tutorial.pdf`

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY EXTENSIBILITY

SPACE BASED COMPUTING SYSTEMS	SUPPORTS EXTENSIBILTY	SPACE BASED COMPUTING SYSTEMS	SUPPORTS EXTENSIBILTY	SPACE BASED COMPUTING SYSTEMS	SUPPORTS EXTENSIBILTY
Apache River		Helios Tuple Space		Network Spaces	
AspectKE*		Heterocera		Open Spaces	
AutoevoSpaces		HTML Page Spaces		Open Wings	M
B-Linda		Info Spaces		P-Linda	
BaLinda K		Jada		P4 Linda	
BaLinda Lisp		Java Spaces		PadSpace	
Bauhaus Linda		Jedi		PoliS	
BISSA		Jini		Prolog-D-Linda v2	
Blitz		JION		PyBrenda	
Blossom		Joyce Linda		PyLinda	
Bonita		JParadise		Ruple	
C-Linda		JXTA Spaces		Semantic Tuple Spaces	
C++ Linda		Kernel Linda	(O)	SecOS	
Comet		Klava		SecSpaces	
Corso		L²imbo	M	SemiSpace	
Crudlet		Lacios		SmallSpaces	
D-Tuples		Lana		SQLSpaces	O
DepSpaces		Law-Governed Infrasturcture		Swarm Linda	
dRuby and Rinda		Law-Governed Linda		Tagged sets	
EgoSpaces		LighTS	C	T-Spaces	(O)
Eiffel Linda		Ligia		TCP Linda	M
eLinda		Limbo		TeenyLIME	
Encrypted Shared Data Spaces		LIME		TIBCO ActiveSpaces	
Entangled		LIME II		The KLAIM family	M
Erlinda	(C)	Limone		TinyLIME	
Fly Object Space		Linda		Triple Space Communication	
Forth-Linda		Lindacap		TuCSoN	
Gaia		Linearizable Byzantine Tuple Space		UML Spaces	
Geo-Linda		LinqSpace		VLOS	
GigaSpaces	S	Linux Tuples		Xcoordination Application Space and Xcoordination Coordination Space	C
Glinda		LuaTs		XMIDDLE	
Globe		LuCe		XML Spaces	O
Grinda		MARS/Moon		XVSM	C,A,M
Gruple		Melinda			
Gspace	R	MobiS			

LEGEND

A	Aspects
C	Coordination
M	Modular
O	Other
R	Replication
S	Scalability

Figure 4.28: Summary Classification of Space Based Computing Systems by Extensibility

4.8 Classification by security

In 2016 one will have a hard time trying to argue that security in a connected world is of no importance. Companies have to ask questions like

- What kind of data do I have to secure?
- From whom needs my data to be secured?
- How can I secure data?

Most of the time the first two questions are pretty straight forward but the answer to the third question is a little bit more tricky to give. This is mainly due to the fact that technologies advance and so solutions normally change over time. Having said this, the commonly accepted approach to secure information involves three key-processes: authentication, authorization, and encryption[87]. Those three processes also find acceptence in the world of space based computing systems and are therefore explained before we start the actual classification process.

- **Authentication** can be simply described as the process of identifying an individual, usually based on a username and password.
- **Authorization** on the other hand enables individuals to access to systems and data based on their identity.
- **Encryption** is the process of transforming data so that it is unreadable by anyone who does not have a fitting decryption key. Commonly known encryption methods are amongst others RSA[88],PGP[89] or SSL[90]encryption.

Figure 4.29 shows which space based computing systems are of interest. They are discussed in the context of security subsequently. Those which are not adressed specifically will be given notice in the summary.

[87] `http://www.bu.edu/tech/security/resources/bestpractice/auth/`
[88] RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman.
[89] PrettyGoodPrivacy
[90] Secure Socket Layer

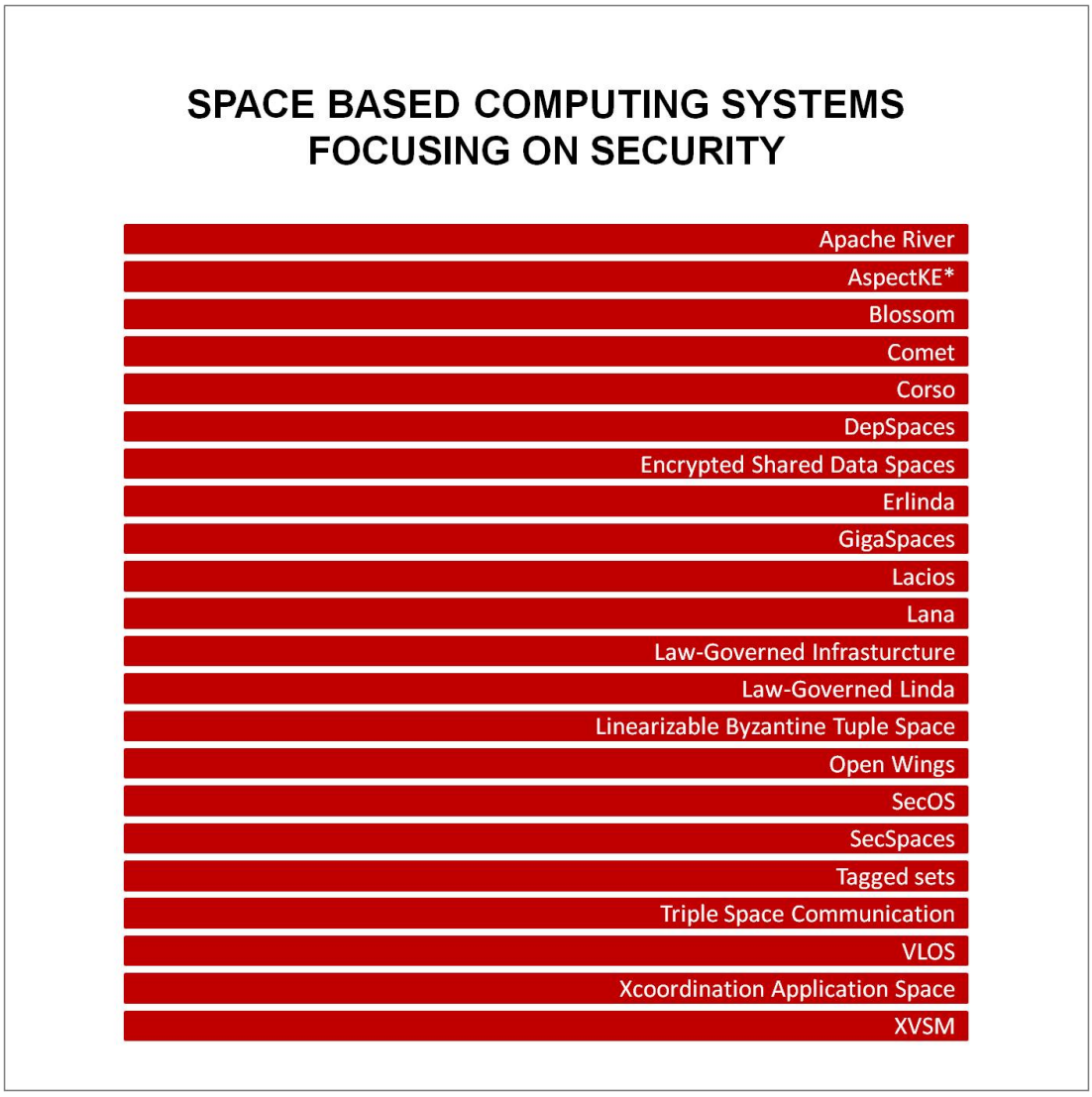


Figure 4.29: Space Based Computing Systems Focusing on Security

AspectKE* [YMA+10] uses aspects to implement different access control policies which focus on the behavior of executing processes. The aspects decide based on their set of rules if a process gets executed or not. Although aspects offer the developer quite a big deal of flexibility the drawback of AspectKE* is that it only can monitor processes and nothing else.
Apache River[91] uses the net.jini.security package to manage the diverse security aspects. The package allows to check if external entities can be trusted and if proxies are authorized to enter a client. In addition the package provides mechanisms to check if code integrity can be granted.
Blossom [GSW97] aims to make tuple spaces more efficient and safer. In short Blossom tries to improve the program performance and the security, *in the sense that the likelihood of program-*

[91] `http://river.apache.org/doc/specs/api/net/jini/security/package-summary.html`

ming errors is reduced.[92] In order to reach this goal Blossom uses strongly typed tuple spaces meaning that *each tuple space is created with a specification that gives the number, order and type of each field of the tuples legally allowed.*[93] In terms of security Blossom offers support for the following topics:

- Blossom uses field access patterns, where tuple fields can only be retrieved by specifying a value, or allowing a wild card variable.
- Blossom also applies access patterns to tuple spaces.
- Blossom allows users to look into tuple spaces, which is referred to as tuple space assertions.

[GSW97] shows that the proposed enhancements help to detect or eliminate parallel programming errors, however there are no performance tests, which indicate how much the user can gain through these measures.

Comet [LP05] offers a global virtual shared-space that can be associatively accessed by all system peers. Such peer groups provide a secure environment where only member peers can access the services available in the respective tuple space. Unsecure peers are connected to the coordinating space via a Proxy (Request Handler). Apart from that there are no other security features.

Corso for example supports authorization mechanisms[94]. In order to access an object a certain authorization process has to be run through. A process has to know the reference to an object in order to act on it. In order to know the reference of an object the process normally owns the object or has to know a password for example. [Sch08a] also explained that users can be managed in a configuration file which defines trusted and allowed users. Trusted users can act as administratos in the space whereas allowed users only can access specified objects.

DepSpaces are also supporting authorization mechanisms [BACF08]. A tuple space in DepSpaces has a single access policy. Whenever a process wants to act on the space it is checked if the process has enough rights - this is done in the so called policy enforcement layer. DepSpace even is said to even work in Byzantine-prone environments.

Encrypted shared data spaces [RDD+08] use encryption schemes, RSA[95] and discrete logarithms, that ensure confidentiality of the data space content in open, possibly hostile, environments.

The big asset of an encrypted shared data space is that it supports tuple matching even over the encrypted data space. The data space does not need to decrypt tuples to perform the search which keeps real content safe from *nosey hosts.*[96] The key management is handled by a fully trusted server which is responsible for all the key-related operations.

[92] [GSW97] page 2.
[93] [GSW97] page 6.
[94] `http://www.sti-innsbruck.at/sites/default/files/D1.2_0.pdf`
[95] Rivest, Shamir and Adleman
[96] [RDD+08] page 15.

The encrypted shared data space itself is used for storing and retrieving plaintext, partially encrypted or completely encrypted tuples. In addition it also performs encrypted searching operations, authenticates valid clients, and safely stores encryption and decryption keys.
The encrypted shared data space offers a different approach to other security mechanisms in space-based computing systems because it does not focus on access control alone.
Erlinda is a parallel computing framework for Erlang. It states that it supports security mechanisms but no specifics can be found in the source.
GigaSpaces is a well-established name in the industry. Therfore it is no suprise that it deals in a professional manner with the topic of security and tries to offer solid and extensible versions. GigaSpaces adresses topics like authentication and authorization. Before even considering if an user is allowed to perform an action upon the space, the user firstly has to undergo the authentication process which is handled by a so called authentication manager. Once an user is cleared the authorization decision manager decides if an user is authorized to execute operations on specific data and services.
GigaSpaces also offers the possibilty of creating an entire authority role model throughout the different system components. Both the authentication layer and the authorization layer offer default implementations, which can be customizedt to he customers needs.[97]
LACIOS [98] [ZBS09] extends Linda and is a data-oriented coordination language which focuses on the design and implementation of multi agent systems used for transportation applications. It also deals with security issues since it wants to guarantee that all agents can exchange data with each other on a global level. In order to do so two levels of security are provided:

- **The global level** is used by the developer who can define a set of rules that are checked when an agent wants to execute any kind of action. Such a rule is normally a boolean expression which can either terminate true or false. If the output is false the request of the agent is rejected.
- **The local level** lets the agents themselves decide if an object poses any kind of threat to them. The agents themselves can define their set of rules which is executed on the locall level.

This mechanisms make it possible to control insertion, perception and retrieval of objects.
Lana [BR02] is based on Java and the Linda tuple space model. One big focus lies on security, specifically on message quality which in a connected world is definitely relevant. Lana uses a so called TrustedZone in order to implement all trusted zone functionality. The trusted zone is the link between the operation system and the device hardware. *This class is used by clients for key operations on the trusted zone, e.g., to set a new profile (setProfile), to specify the evidence required for an incoming message to be validated (setRequiredEvidence), as well as to specify the profile certificates required when validating message quality (addRequiredCertificate).*[99] This possibilities definitely offers quite an advantage when agents move through the web.

[97] `http://wiki.gigaspaces.com/wiki/display/XAP9/Security`
[98] Language for Agent Contextual Interaction in Open Systems
[99] [BR02] page 77.

Law-Governed Linda[100] [ML95] extends the Linda model with law-governed architectures in order to enhance the security of tuple spaces. To reach this goal Law-Governed Linda forces all processes to adopt a set of specific rules which are monitored by a controller who has a copy of the law that applies to the specific tuple/tuple space. Only if a process fulfills all the laws it can execute operations on the tuple/tuple space. A relative of LGL is **Law-Governed Infrastructure**[101] [MMU01]. It allows agents and processes to communicate with each other under a certain policy, which is called the law of the group. Policies exist for security issues as well as for coordination issues.
LGI characteristics are:

- Laws under LGI are sensitive to the content of the tuples being handled.
- Laws are sensitive to the state of agents.
- Enforcement of laws can occur at either the client, the server, or anywhere in the network.
- LGI supports different levels of security.

Every agent can join an arbitrary number of groups and follows their different law as necessary. A law in LGI can be described as a reactive rule which executes an operation on a certain event. Like in LGL, a controller monitors everything to secure that policies are fulfilled. At the moment no cryptographic techniques are used to enhance security which can be seen as a drawback.
Secure Object Spaces[102] [VBO03], [BDN02] extends Linda with fine-grained access control based on locking in order to strengthen the security of space-based computing systems without being too restrictive. A lock can be seen as a specific value that represents the key to a given tuple. SecOS offers a set of different variants for the locking process:

- Symmetric key locking: One key locks and unlocks the tuple.
- Asymmetric key locking: A public key locks the tuple and a private key can unlock it.
- Fine grained access control at fields and tuple level.

Ideas of SecOS have influenced also **SecSpaces** [BGLZ02], which is a Linda-like coordination model that supports secure data-driven coordination in open environments.
SecSpaces introduces two new fields which can both add value to the coordination model:

- Partitioning fields: The tuple space partitioning is achieved through the introduction of a partition field in the tuples. An agent can only enter the field if it knows the name of the partition. This partitioning mechanism avoids all agents having the same view on the data contained in a tuple space.

[100] LGL
[101] LGI
[102] SecOS

- Cryptographic fields: The cryptographic fields can distinguish between agents who can only execute read operations and those who can only execute take operations through the use of asymmetric cryptography.

SecSpaces offer a good approach to secure tuple spaces and tuple fields and therefore they are quite popular in the scientific community.
Tagged sets [OH05] are a virtual shared memory approach *that relies on tags based on propositional logic to lock and select values*[103] with the aim to support scenarios like shared data repositories, message passing or publish/subscribe algorithms. Tagged sets provide good security because tags are usually associated with a key that defines the protection grade and only if a user presents his key he is allowed to execute an operation on the tag.
VLOS[104] is presented in [MCW02] and [CM03] and can be described as a distributed operating system based on tuple spaces. Security in VLOS is granted through a capability-based approach. Capabilities in VLOS normally consist of a unique identifier, the object type and the name of the tuple space with which the object is associated as well as of a set of rights, a cryptographic and a hash function [UWJ07].
In the previous section the use of aspects in **XVSM** has been already discussed - since aspects give the programmer the freedom to extend the space with own code sniplets they are also interesting in the context of security, where they can be used for implementing authentication and monitoring processes amongst others [Bar10].
Another interesting space based computing system which was not part of the entire qualification process is **CryptoKlava** which provides the cryptography which can be used in the combination with Klava [BDNP01]. CryptoKlava uses the Java Cryptography Extension[105], a set of packages that provide a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms.

4.8.1 Summary

Figure 4.30 summarizes this section. What can be seen is that only a few space based computing systems are focusing on cryptograpic topics whereas quite a few focus on authentication and authorization methodologies. GigaSpaces is probably the most technically mature space based computing system in terms of security but there are other interesting space based computing systems like XVSM an AspectKE* which make use of the aspects which offers great versatility to the developer. The Klaim family has also different approaches dealing with security as well as Linearizable byzantine fault-tolerant tuple space [106] [NBCdSFCL07], [Bes06] which use simple quorum-based protocols and more complicated operations use consensus-based protocols to enhance fault-tolerance. Summing up this topic, one can also see that in this area more needs to be done, since only one fifth of all space based computing systems is dealing with security matters.

[103] [OH05] page 15.
[104] Virtually Linda Operating System
[105] JCE
[106] LBTS

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS BY SECURITY

SPACE BASED COMPUTING SYSTEMS	SUPPORTS SECURITY	SPACE BASED COMPUTING SYSTEMS	SUPPORTS SECURITY	SPACE BASED COMPUTING SYSTEMS	SUPPORTS SECURITY
Apache River	AU, AZ	Helios Tuple Space		Network Spaces	
AspectKE*	AU, AZ	Heterocera		Open Spaces	
AutoevoSpaces		HTML Page Spaces		Open Wings	AU, AZ
B-Linda		Info Spaces		P-Linda	
BaLinda K		Jada		P4 Linda	
BaLinda Lisp		Java Spaces		PadSpace	
Bauhaus Linda		Jedi		PoliS	
BISSA		Jini		Prolog-D-Linda v2	
Blitz		JION		PyBrenda	
Blossom	AU, AZ	Joyce Linda		PyLinda	
Bonita		JParadise		Ruple	
C-Linda		JXTA Spaces		Semantic Tuple Spaces	
C++ Linda		Kernel Linda		SecOS	AZ
Comet	(AZ)	Klava (Crypto Klava)	C	SecSpaces	AU, AZ
Corso	AZ	L²imbo		SemiSpace	
Crudlet		Lacios	AU, AZ	SmallSpaces	
D-Tuples		Lana	AZ	SQLSpaces	
DepSpaces	AZ	Law-Governed Infrasturcture	AZ	Swarm Linda	
dRuby and Rinda		Law-Governed Linda	AU, AZ	Tagged sets	AU, AZ
EgoSpaces		LighTS		T-Spaces	
Eiffel Linda		Ligia		TCP Linda	
eLinda		Limbo		TeenyLIME	
Encrypted Shared Data Spaces	AU, AZ, C	LIME		TIBCO ActiveSpaces	
Entangled		LIME II		The KLAIM family	AU, AZ
Erlinda	(AU, AZ)	Limone		TinyLIME	
Fly Object Space		Linda		Triple Space Communication	AU, AZ
Forth-Linda		Lindacap		TuCSoN	
Gaia		Linearizable Byzantine Tuple Space	O	UML Spaces	
Geo-Linda		LinqSpace		VLOS	AU, AZ, C
GigaSpaces	AU, AZ, C	Linux Tuples		Xcoordination Application Space and Xcoordination Coordination Space	AU, AZ
Glinda		LuaTs		XMIDDLE	
Globe		LuCe		XML Spaces	
Grinda		MARS/Moon		XVSM	AU, AZ
Gruple		Melinda			
Gspace		MobiS			

LEGEND	
AU	Authentication
AZ	Authorization
C	Cryptography
O	Other

Figure 4.30: Summary Classification of Space Based Computing Systems by Security

4.9 Classification by life cycle management

The efficient management of data has become more important to businesses in the last years, since organizations must deal not only with bigger amounts of data but also have to handle some

information in a special[107] way. Therefore efficient life cycle management should manage data from its creation and *initial storage to the time when it becomes obsolete and is deleted.*[108] When it comes to space based computing systems especially the last part is of importance since it keeps the amount of data within the spaces on an acceptable level and prevents the space from growing too big for no reason.
In 1997 [MW97] already explained in their paper why garbage collection is needed in space based computing systems. They also describe the process of garbage collection as a two-step process, namely searching data which can be deleted and after that the actual deleting process. One can use different alghorithms in order to decide if certain data in a space is still useful. Subsequently I will write about two simple garbage collection algorithms [JHM11] to give a better understanding of the topic itself.
The mark-and-sweep garbage collection algorithm was the first garbage collection algorithm to be developed by John McCarthy around 1959. The mark-and-sweep algorithm is a so called a tracing garbage collector because it runs through every object/data that is somehow connected with the space. Data that can be entered directly is a so called root and represents the starting point for every search. All objects that are somehow accesible from a root are a marked as alive and are considered as no garbage. On the other hand all objects that can not be reached are considered as garbage and therefore can be thrown away. Figure 4. 33[109] illustrates how the mark and sweep alghoritm works.
Before the garbage collection process we see six objects all not marked yet, and one root. The arrows show the graphs that connects the objects. In the mark phase of the algorithm all lobjects that could have been reached from the root are marked TRUE. In this case the objects 1,2,3, and 6. After the sweep phase the objects 4 and 5 have been deleted since they have not been marked before.

[107] E.g. employee relevant data has to be stored seven years in Austria
[108] `http://searchstorage.techtarget.com/definition/data-life-cycle-management`
[109] `http://www.brpreiss.com/books/opus5/html/page424.html`

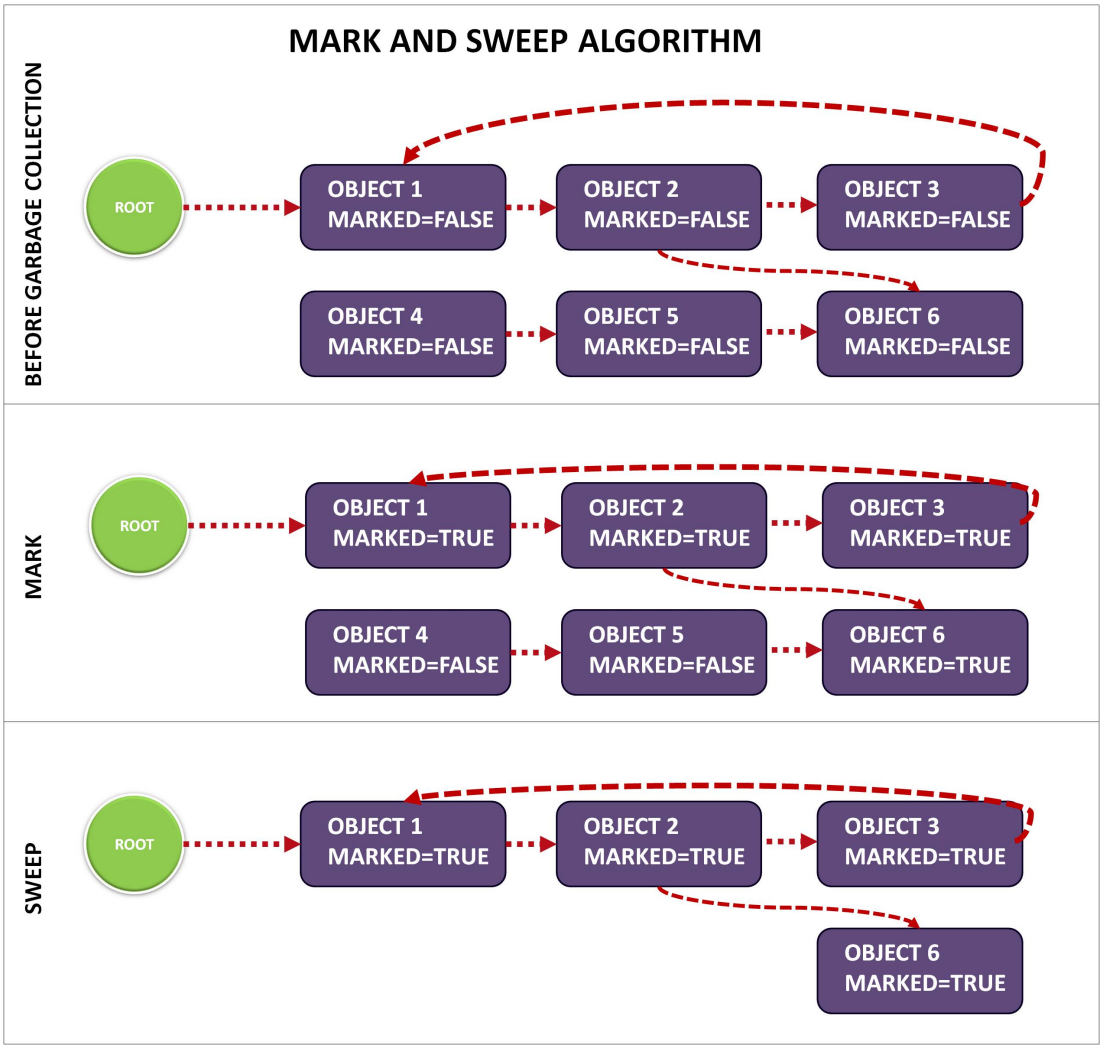


Figure 4.31: Mark and Sweep Algorithm

Another way to identify garbage/unused objects is reference counting. Every object is given a reference count, which indicates the number of other objects pointing to this object [MW97]. As long as the object's reference count is higher than zero, other objects are pointing to it and it is classified as no garbage. When the object count reaches zero it can safely be deleted.

[MW97] shows that the mark and sweep algorithm can be used also in tuple spaces to mark unused processes, data or tuple spaces.

Since that time a lot has happened in life cycle management but only a few space based computing implementations are seriously dealing with this topic. Subsequently the most relevant space based computing systems with the focus on garbage collection will be discussed.

Ligia [MW98] extends the Linda tuple space and is a Java-based implementation which includes garbage collection of tuple spaces and agents. Garbage collection is done with a mark and sweep algorithm that once starts at the root to mark agents/tuplespaces as active and a second time the

algorithm starts at the end of marked agents/tuplespaces to see if it is possible to reach the root. Only agents/tuplespaces that have not been marked in neither of the two phases are considered as garbage and therfore can be deleted. In order to make garbage collection possible in the first place, the server needs to be aware of the agents. In Ligia this is done by two simple methods. The moment an agents wants to create an object it has to interact with the server who adds an unique name to the agent which can be monitored. When an agent leaves it has to do it with the leave method provided by Ligia so that the server knows that the agent is no longer active. However if the user is sloppy and does not use this method the server will consider the agent as activ although it is not. This of course makes it difficult for the garbage collector to work efficiently. However, the big issue with Ligia is that one can only delete tuple spaces or agents but no tuples. [UWJ07] stated that the main problem in introducing garbage collection for tuples is the lack of suffcient information about the tuples themselves,e.g. how they are connected with each other and if they are active or not.The reason for that is that whilst tuple spaces have unique IDs which can be referenced tuples are normally addressed by the values they are holding.
Lindacap uses multicapabilities to overcome this issue. Such a multicapability consists of a unique tag to differentiate between different capabilities for the same template [UWJ07]. And now it is possible to understand which agent knows about which tuples. In Lindacap garbage collection is only done under specific circumstances, e.g. if there is not enough memory space available. Garbage collection mainly is not a routine process because it comes at the price of needing quite a bit of system resources.
Also **XVSM** and **TSC** support garbage collection. The Triple Space API for example acts as a garbage collector, removing ”hanging” operations or checking for finished operations that might be of no use anymore. Since these three space based computing systems are of the same family - they follow more or less the same ideas.
Another space based computing system which claims to support garbage collection is **PyLinda**[110] but no closer information in how it supports it could be found.
To sum up this chapter one has to say that not much information can be found concerning garbage collection although it is of some importance for an efficient and optimized space based computing system.

[110] `http://code.google.com/p/pylinda/`

CHAPTER 5

Conclusion

5.1 Summary

This thesis has tried to bring a certain amount of structure to the different space based computing systems, with the aim to bring transparancy to their general focus points, strenghts and weaknesses.
Analysing the different space based computing systems has shown that the variety and the different stages of development of the single space based computing systems make it difficult to compare them with each other. However, the thesis can give the reader an overview of the individual systems as well as their focus areas.
Figure 5.1 at the end of this chapter will show a complete overview of the gathered results.
Before that I will summarize what has been done in this paper. In chapter 2 I have focused on the historical aspects of space based computing systems, when they have been first introduced and how long they have been of relevance. Also a first overview of their main focus areas has been given. After that a few space based computing systems like Linda, XVSM, TuCSoN, LIME et cetera have been presented in more detail to give the reader a general understanding about how space based computing systems work. After that different application scenarios for space based computing systems have been introduced in chapter 3, in order to give the reader an idea why this field of research could be of importance. Chapter 4 focused on the main classification process and it has been possible to classify the 103 space based computing systems after the following themes: families, operations, coordination concepts, space substructures, data types, extensibility, security and life cycle management. Since life cycle management is not a very popular topic in space based computing systems and most programming lanuages offer already possibilities for garbage collection, this section of chapter 4 seems relatively unfinished since one might think there is more to write about. However, understanding that garbage collection in space based computing systems is not trivial only a few space based computing systems really took a closer look at this problem.

Three topics have been left out in the classification process: architecture[1], scalability and performance. This decision was made due to various reasons.
Classifying space based computing systems after their system architecture would mean that one has to classify space based computing systems into two groups: in spaces based computing systems that are embedded[2] and in those which are standalone[3] [Sch08a]. Since embedded systems can also function as standalone systems it didn't make much sense to write in detail about this topic, since the system architecture probably is not the key criteria that a developer uses when deciding which space to use. Having said this, it is important to know that embedded systems like XVSM are lighter and normally easier to use because they are hidden within the application. Also network architecture did not seem to have the importance to be classified in all detail. Most space based computing systems have a centralized structure whereas the smaller group is decentralized. Also a few spaces support no remote communication at all. The information about system and network architecture however is represented in figure 5.1.
Scalability and performance are not included in the classification process since these two parts can be seen as quantitative indicators and since not all space based computing systems have a proper implementation to compare them with each other in a fair way, they have been excluded from this classifcation. However [Lwe10] and [OG02] have done intersting research in this field. Figure 5.1 includes a classification section on those spaces which say that they aim for scalabilty. However, these sections might not be complete.
In my opinion there is no such a thing as the perfect space based computing system. However, there are certain space based computing systems which are more advanced than others when it comes to certain needs.
When it comes to flexibility, ease of use and possibilities offered to the developer XVSM and its family members are in my opinion hard to beat. It offers much more possibilities than any other system, which is mainly because of its containers which allow for different coordination types and the use of aspects which easily can integrate topics like security, replication or anything the developer would like to add.
However if a company would like to use a space based computing system, I immediately would suggest to go for GigaSpaces which is a solid commercial solution that guarantess updates, support and the steady development of the system itself. Academic solutions often pose a risk since a company can not be sure how long support for the space based computing system can be assured.
If I would have an application scenario that focuses on mobility either a member of the LIME family or TuCSoN would be top on my list, since they are well tested, well described and pretty easy to use. And although a member of the Klaim family would be probably as good if not even better, the only drawback is that Klaim at the beginnig is not so easy to understand and not as well described as the other two options.
When it comes to security XVSM is a good solution since it does not only focus on this one aspect, neglecting all the other important issues on the other hand GigaSpaces also offers solid solutions for security issues. So here the price would be probably the key criteria for a decision.

[1] system as well as network architecture
[2] The space based computing system is directly integrated in the application.
[3] The space based computing system runs completely on its own

This part of the conclusion consolidates all the found information in figure 5.1. In the glossary all the 103 space based computing systems of the survey are described shortly and references to useful papers and websites are given. The second part of the conclusion will focus on the most recent developments which have not been included in the survey because their release was too late for inclusion.

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS A-J

SPACE BASED COMPUTING SYSTEMS	FAMILY AND GENERAL INFORMATION	OPERATIONS	COORDINATION	SUBSTRUCTURE	DATA TYPE	EXTENSIBILITY	ARCHITECTURE	SECURITY	LIFECYCLE MANAGEMENT	SCALABILITY
Apache River	SBC,JS,OS,F	B+NB,N,T	L,FIFO	O	O		E,S,C	AU,AZ		SUPPORTED
AspectKE*	SBC,JS,OS,F	B+E+NB+X	L		T		E,S,C	AU,AZ		
AutoevoSpaces	SBC,JS,C,N	B+NB+X	L		O		S,C			
B-Linda	SBC,L,OI	B+E+X	L		T		S,C			
BaLinda K	SBC,L,OI	B+X	L		T		S,C			
BaLinda Lisp	SBC,L,OI	B+X	L		T		C			
Bauhaus Linda	SBC,L,OI	B	L		T		S,C			
BISSA	SBC+P2P, L+JS+FS,OS,F	B+NB,N,T	L	S	T		E,S,C,D			SUPPORTED
Blitz	SBC,JS,OS,F	B+NB+X,N,T	L,FIFO	O	O		E,S,C,D			
Blossom	SBC,L,OS,OI	B+E+NB+X	FIFO	O	T		S,C	AU,AZ		SUPPORTED
Bonita	SBC,L,OI	B+X	L		T		C			
C-Linda	SBC,L,OS+C,F	B+E+NB+X	L		T		S,C			
C++ Linda	SBC,L,OI	B+E+X	L		P, S		E,S,C			
Comet	SBC+P2P,OI	B,N	L	S	P		S,C,D	(AZ)		SUPPORTED
Corso	SBC,L+VSM+FS OS+C,F	S,N,T	O	O,S	P		S,C,D	AZ	SUPPORTED	
Crudlet	SBC,JS,OS,F	B+NB+X,N,T	L,FIFO	O	O		E,S,C			
D-Tuples	SBC,L+FS,OI	B,N,T	H	O	T		E,S,C			
DepSpaces	SBC,FS,OS,F	B+NB	L		O		E,S,D	AZ		
dRuby and Rinda	SBC,L+FS,OS,F	B	L		O		E,S,D			
EgoSpaces	SBC,FS,OS,F	B+NB+X,N,T	L		T		E,S,D			
Eiffel Linda	SBC,L,OI	B+E	L		T		C			
eLinda	SBC,L,OI	B+X	L,FIFO		T		C			SUPPORTED
Encrypted Shared Data Spaces	SBC,OI	B	L,K		T			AU,AZ,C		
Entangled	P2P,FS,OS,F	B	L,H		T		S,D			
Erlinda	SBC,JS,OS,F	B+E+NB,N,T	L,FIFO		O	(C)	S,C,D	(AU,AZ)		SUPPORTED
Fly Object Space	SBC,JS+FS, OS+C,F	B,N,T	L	O	O		S,C,D			
Forth-Linda	SBC,L,OI	B+E	L		T		S,C			
Gaia	SBC,FS,OS,F						E,S,D			
Geo-Linda	SBC,L+FS,OI	B+X,N	L	O	T		S,C			
GigaSpaces	SBC,JS,C,F	B+NB+X,N,T	L, FIFO,K	O	S	S	E,S,C,D	AU,AZ,C		SUPPORTED
Glinda	SBC,L,OI	N,T	L	O	P, T		S,C			
Globe	SBC,JS,OI	B+NB	L,FIFO		T		C,D			SUPPORTED
Grinda	P2P+FS,OS,F	B+X,N,T	L	O	O		E,S,C,D			
Gruple	SBC,JS+FS,OS,F	B,N,T	L	O	O		E,S,D			
Gspace	SBC,FS,OI	B	L	S	S	R	S,C			
Helios Tuple Space	SBC,L,OI	B+E+NB+X	L		T					SUPPORTED
Heterocera	SBC,L+FS,OS,F	B	L		T		C			
HTML Page Spaces										
Info Spaces	SBC+P2P,JS,OI	S					E,S,D			
Jada	SBC,L+FS,OI	B	L,FIFO	O	O		S,C,D			
Java Spaces	SBC,L+JS, OS+C,F	B+NB+X,N,T	L,FIFO	O	O		S,C			SUPPORTED
Jedi	SBC,FS,OI	S,N,T	L	O	T		S,D			
Jini	SBC,JS,OS,F	B+NB+X,N,T	L,FIFO	O	O		E,S,C			SUPPORTED
JION	SBC,JS,OS,F	B,N,T	L		T		S,C			
Joyce Linda	SBC,L,OI	B+E+NB	L		T					
JParadise	SBC,L,C,F	B+NB+X	L		T		S,C			SUPPORTED
JXTA Spaces	SBC+P2P,OS,F	B+X,N,T	L,FIFO	O	T		S,C,D			

LEGEND

FAMILY AND GENERAL INFORMATION
- SBC — Influenced by the Space Based Computing Paradigm
- SW — Influenced by the Semantic Web Paradigm
- P2P — Influenced by the P2P Paradigm
- L — Space Based Computing System based on Linda
- JS — Space Based Computing System based on JavaSpaces
- VSM — Space Based Computing System Based on VSM and DM
- FS — Free Space Based Computing System
- OS — Opens Source
- C — Commercial
- N — No Implementation and no Information available
- OI — Only Information available
- F — Implementation and no Information available

OPERATIONS
- B — Basic Linda Operations
- E — Eval Operation
- NB — Non-Blocking Operations
- X — Extendend Operations
- S — Speciality
- N — Notifications
- T — Transactions

COORDINATION
- L — Linda Coordination
- FIFO — First-In-First-Out
- LIFO — Last-In-First-Out
- R — Random
- K — Keys
- H — Hashtables
- O — Other

SUBSTRUCTURE
- O — Ordering mechanism
- S — Substructure

DATA TYPE
- O — Objects
- T — Tuples
- P — Primitives
- S — Specialities

EXTENSIBILITY
- A — Aspects
- C — Coordination
- M — Modular
- O — Other
- R — Replication
- S — Scalability

ARCHITECTURE
- E — Embedded
- S — Standalone
- C — Centralized
- D — Decentralized

SECURITY
- AU — Authentication
- AZ — Authorization
- C — Cryptography
- O — Other

Figure 5.1: Classification of Space Based Computing Systems A-J

SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS K-S

SPACE BASED COMPUTING SYSTEMS	FAMILY AND GENERAL INFORMATION	OPERATIONS	COORDINATION	SUBSTRUCTURE	DATA TYPE	EXTENSIBILITY	ARCHITECTURE	SECURITY	LIFECYCLE MANAGEMENT	SCALABILITY
Kernel Linda	SBC,L,OI	B+NB	L		T	(O)	S,C			
Klava	SBC,JS,OS,F	B+E+X,N,T	L,H		O		E,S,D	C		
L²imbo	SBC,L,OI	B	L	S	T	M	S,C			
Lacios	SBC,L,OI	B+X,N	L		T		D	AU, AZ		
Lana	SBC,L,OI	B,N	L	O	T		E,S,D	AZ		
Law-Governed Infrasturcture	SBC,L,OI	B	L		T, O			AZ		
Law-Governed Linda	SBC,L,OI	B	L		T, O			AU, AZ		
LighTS	SBC,L,OS,F	B+NB+X	L		T, S	C	E,S			
Ligia	SBC,L,OI	B+NB+X	L	S	T				SUPPORTED	
Limbo	SBC,L,OI	B	L	S	T					
LIME	SBC,L,OS,F	B+NB+X,N,T	L	S	T		E,S,C			
LIME II	SBC,L,OS,OI	B+NB+X,N,T	L	S	T		E,S,C			
Limone	SBC,L,OS,F	B+NB+X,N,T	L,H	S	T		E,S,C			
Linda	SBC,L,OS+C,F	B+E+NB	L		T		S			SUPPORTED
Lindacap	SBC,L,OI	B	L,K		T				SUPPORTED	
Linearizable Byzantine Tuple Space	SBC,L+FS,OI	(B),N	L,FIFO		T		D	O		
LinqSpace	SBC+P2P, JS+VSM,OI	B+NB,N,T		O,S	S		E,S,D			
Linux Tuples	SBC,L,OS,F	B+NB+X	L		T		D			
LuaTs	SBC,L,OI	B+NB+X,N,T	L		T		E,S,D			
LuCe	SBC,L+FS,OI	B+NB,N,T	L		T		E,S,D			
MARS/Moon	SBC,L+JS+FS, OS,F	B,N,T	L,FIFO	O	O		E,S,C			
Melinda	SBC,L,,OI	B+E+NB	L		T					SUPPORTED
MobiS	SBC,L,OI	B	L	O,S	T		E,S			SUPPORTED
Network Spaces	SBC,L,C,F	B+NB	L		T		E,S,C			
Open Spaces	SBC,L+JS+FS,OI	B+NB+X	L		T		E,S,C,D			
Open Wings	SBC,JS,OS,F		L	O	O	M	E,S,D	AU, AZ		SUPPORTED
P-Linda	SBC,L,OS,F	,B+E+NB,T	L		T					
P4 Linda	SBC,L,OS,F	B+E	L		T					
PadSpace	SBC,L,OI						E,S,D			
PoliS	SBC,L,OI	B+E+NB	L	O,S	T					
Prolog-D-Linda v2	SBC,L,OS,F	B+E+NB	L		T		E,S,C			
PyBrenda	SBC,L,OS,F	B+(NB)+X	L		T					
PyLinda	SBC,L,N	B+(NB)+X	L		T				SUPPORTED	
Ruple	SBC,FS,C,OI	B+X	L		O		E,S,C			
Semantic Tuple Spaces	SBC+SW,JS,OI	B+NB,N,T	L	S	S					
SecOS	SBC,L,OI	B+NB	L,H		T		D	AZ		
SecSpaces	SBC,L,OI	B+NB,T	L		T		D	AU, AZ		
SemiSpace	SBC,JS,N	B,N,T	L				C			SUPPORTED
SmallSpaces	SBC,L,N									
SQLSpaces	SBC,L,OS,F	B+NB+X,N,T	L		T	O	E,S,C			
Swarm Linda	SBC,L,OI	B	L	S	T					SUPPORTED

LEGEND

FAMILY AND GENERAL INFORMATION

SBC	Influenced by the Space Based Computing Paradigm
SW	Influenced by the Semantic Web Paradigm
P2P	Influenced by the P2P Paradigm
L	Space Based Computing System based on Linda
JS	Space Based Computing System based on JavaSpaces
VSM	Space Based Computing System Based on VSM and DM
FS	Free Space Based Computing System
OS	Opens Source
C	Commercial
N	No Implementation and no Information available
OI	Only Information available
F	Implementation and no Information available

OPERATIONS

B	Basic Linda Operations
E	Eval Operation
NB	Non-Blocking Operations
X	Extendend Operations
S	Speciality
N	Notifications
T	Transactions

COORDINATION

L	Linda Coordination
FIFO	First-In-First-Out
LIFO	Last-In-First-Out
R	Random
K	Keys
H	Hashtables
O	Other

SUBSTRUCTURE

O	Ordering mechanism
S	Substructure

DATA TYPE

O	Objects
T	Tuples
P	Primitives
S	Specialities

EXTENSIBILITY

A	Aspects
C	Coordination
M	Modular
O	Other
R	Replication
S	Scalability

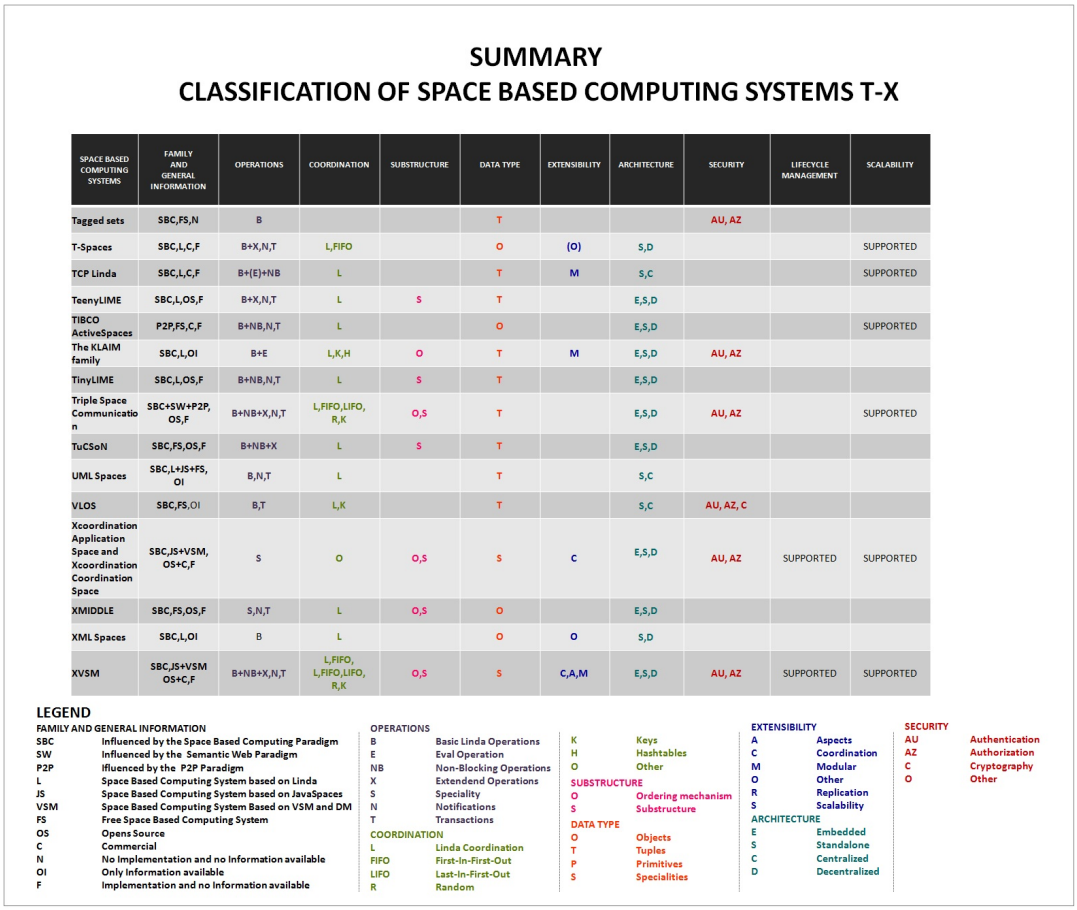
ARCHITECTURE

E	Embedded
S	Standalone
C	Centralized
D	Decentralized

SECURITY

AU	Authentication
AZ	Authorization
C	Cryptography
O	Other

Figure 5.2: Classification of Space Based Computing Systems K-S



SUMMARY
CLASSIFICATION OF SPACE BASED COMPUTING SYSTEMS T-X

SPACE BASED COMPUTING SYSTEMS	FAMILY AND GENERAL INFORMATION	OPERATIONS	COORDINATION	SUBSTRUCTURE	DATA TYPE	EXTENSIBILITY	ARCHITECTURE	SECURITY	LIFECYCLE MANAGEMENT	SCALABILITY
Tagged sets	SBC,FS,N	B			T			AU, AZ		
T-Spaces	SBC,L,C,F	B+X,N,T	L,FIFO		O	(O)	S,D			SUPPORTED
TCP Linda	SBC,L,C,F	B+(E)+NB	L		T	M	S,C			SUPPORTED
TeenyLIME	SBC,L,OS,F	B+X,N,T	L	S	T		E,S,D			
TIBCO ActiveSpaces	P2P,FS,C,F	B+NB,N,T	L		O		E,S,D			SUPPORTED
The KLAIM family	SBC,L,OI	B+E	L,K,H	O	T	M	E,S,D	AU, AZ		
TinyLIME	SBC,L,OS,F	B+NB,N,T	L	S	T		E,S,D			
Triple Space Communication	SBC+SW+P2P, OS,F	B+NB+X,N,T	L,FIFO,LIFO, R,K	O,S	T		E,S,D	AU, AZ		SUPPORTED
TuCSoN	SBC,FS,OS,F	B+NB+X	L	S	T		E,S,D			
UML Spaces	SBC,L+JS+FS, OI	B,N,T	L		T		S,C			
VLOS	SBC,FS,OI	B,T	L,K		T		S,C	AU, AZ, C		
Xcoordination Application Space and Xcoordination Coordination Space	SBC,JS+VSM, OS+C,F	S	O	O,S	S	C	E,S,D	AU, AZ	SUPPORTED	SUPPORTED
XMIDDLE	SBC,FS,OS,F	S,N,T	L	O,S	O		E,S,D			
XML Spaces	SBC,L,OI	B	L		O	O	S,D			
XVSM	SBC,JS+VSM OS+C,F	B+NB+X,N,T	L,FIFO, L,FIFO,LIFO, R,K	O,S	S	C,A,M	E,S,D	AU, AZ	SUPPORTED	SUPPORTED

LEGEND

FAMILY AND GENERAL INFORMATION
- SBC — Influenced by the Space Based Computing Paradigm
- SW — Influenced by the Semantic Web Paradigm
- P2P — Influenced by the P2P Paradigm
- L — Space Based Computing System based on Linda
- JS — Space Based Computing System based on JavaSpaces
- VSM — Space Based Computing System Based on VSM and DM
- FS — Free Space Based Computing System
- OS — Opens Source
- C — Commercial
- N — No Implementation and no Information available
- OI — Only Information available
- F — Implementation and no Information available

OPERATIONS
- B — Basic Linda Operations
- E — Eval Operation
- NB — Non-Blocking Operations
- X — Extendend Operations
- S — Speciality
- N — Notifications
- T — Transactions

COORDINATION
- L — Linda Coordination
- FIFO — First-In-First-Out
- LIFO — Last-In-First-Out
- R — Random

- K — Keys
- H — Hashtables
- O — Other

SUBSTRUCTURE
- O — Ordering mechanism
- S — Substructure

DATA TYPE
- O — Objects
- T — Tuples
- P — Primitives
- S — Specialities

EXTENSIBILITY
- A — Aspects
- C — Coordination
- M — Modular
- O — Other
- R — Replication
- S — Scalability

ARCHITECTURE
- E — Embedded
- S — Standalone
- C — Centralized
- D — Decentralized

SECURITY
- AU — Authentication
- AZ — Authorization
- C — Cryptography
- O — Other

Figure 5.3: Classification of Space Based Computing Systems T-X

5.2 Latest developments in the area of space based computing systems

The second part of the conclusion presents space based computing systems that have not been included in the survey because their release was too late for inclusion.
I started to work on this thesis in 2013 and had found 103 space based computing systems by 2014. During the time I was working on the classification process between 2014 and 2015, already existing space based computing systems have been extended or improved and new space based computing systems have been introduced.
Being aware of this fact I always planned to go back where I started in the beginning and look at the most recent developments. I always felt that getting as much unbiased information as possible is an important factor also for developers because it helps them to facilate the actual knowledge building.
So the final part of this thesis will present a completely new space based computing system as well as extions of already existing systems.

MELON[4] is a practical approach for distributed communication in MANET applications written in Ruby and was introduced by [CB14]. It provides *persistent messages, reliable FIFO-ordered multicast, effcient bulk retrieval, and simple message streaming.*[5] In his PhD thesis [Col14] gives an in detail description of MELON with detailed examples and benchmarks. MELON uses the idea of a distributed shared tuple space, and enhances it with the following specifics:

- MELON divides the messages within the space in two groups: a remove-only groupl, and a read-only group. Remove-only messages can only be retrieved once and must be removed when retrieved. Read-only messages may never be explicitly removed, only be copied from the message store.
- Messages can be implemented as any structure which can be matched by a template.
- MELON adresses the storage limitations of mobile devices by allowing messages to be automatically garbage collected.
- In MELON all retrieval operations are limited to a best effort approach.

For me the it was very interesting that also had problems to compare MELON in a quantitative manner:
We found most projects were based on traditional distributed computing paradigms, but it was not possible to assert any conclusions regarding the underlying paradigms, since the project implementations compared used differen languages and were of varying quality. In order to study the paradigms performance in a quantitative manner, we implemented our own versions of three commonly-used paradigms (publish/subscribe, RPC, and tuple spaces) with as much shared code as possible. This allowed us to fairly compare the paradigms using real applications.[6]
It is the same hurdle that I found when I was thinking over methodolgies on how one could classify quantitive aspects of the diverse space based computing systems. I also came to the conclusion that you would have to create similar conditions to make the space based computing systems comparable. In my opinion this can not not be done in a general way but has do be done in a more specific way, where you take application scenarios and expectations on what goal you want to achieve into account.
When I looked at the newer material I found a few ideas that actually where combining already existing space based computing implementations with other technologies. Buzz [PLBB15], a programming language for heterogeneous robot swarms, is such an idea. The main idea is that Buzz offers *primitives to define swarm behaviors both from the perspective of the single robot and of the overall swarm.*[7] In order to enable virtual stigmergy[8] a distributed tuple space analogous to Linda is used. With Buzz a second representative after SWARMLINDA shows that space based computing systems can facilitate swarm behaviour.

[4] Message Exchange Language Over the Network
[5] [CB14] page 1.
[6] [Col14] page 118.
[7] [BDL+14] page 1.
[8] Virtual stigmergy is a data structure that allows a swarm of robots to share data globally.

A completely different application scenario is adressed by Cwmwl [FW14]. Cwmwl is a Platform-as-a-Service cloud environment that combines the LINDA coordination language, an in-memory key-value store, with functional programming to facilitate efficient execution of tenant plugins and applications. In the implementation a tuple space plays a central role in introducing deterministic services for basic parallel programming, including message passing, persistent infinite message pools and transactions. [FW14] show in their paper that using a tuple space can enhance the capability of a PaaS framwork.
What I also found out during my most recent research is that space based computing implementations that are already well established focus on constantly developing new features or try to explore new application scenarios. The Peer Model which is described in [KCH14] and [Rau14] is a good example. The Peer Model enables scalability, compsability, the separation of coordination and business logic. It adresses that fact space-based frameworks have a reputation to be hard-to-use which often leads developers to decide against them.
The Peer Model is a programming model for the design of coordination strategies among multiple nodes, aiming to bridge design and implementation.[9] [KCH14] presents an application scenario from the railway domain in which embedded nodes detect approaching trains and route this information over several forwarder nodes to the level crossing. In order to secure a good scalability and decoupling a space based computing middleware is used where communication is handled through the space. The Peer Model is compared with other spaced based computing systems such as TeenyLime or LINC and shows good results in the sectors of flexibilty and extensibility. The Peer model offers a good abstraction for communication and coordination. Furthermore it generates source code automatically. In the long term these qualities can help to enhance the development process of embedded systems.
[Rau14] presents a PeerSpace.NET framework that is based on the Peer Model, is implemented in .NET and based on Xcoordination AppSpace. The implementation focuses on an usable API and advanced error handling possibilities. The thesis of [Rau14] also shows that the usability of the PeerSpace's API is much better than the usability of the WCF's API.
Also a few interesting thesis written from students of the Space Based Computing Group of the Institute of Computer Languages at the Vienna University of Technology have been presented in 2014 and 2015. [Wat14] introduced different lifecycle and memory management ideas for MozartSpaces, the Java reference implementation of eXtensible Virtual Shared Memory (XVSM). [Wat14] introduces an event-driven architecture which also provides time-based events, as well as an additional extensibility mechanism - jobs, which will support the execution of arbitrary code in reaction to events. *Building on these two mechanisms, a leasing mechanism for managing the lifetime and validity of data in the space is implemented.*[10] The leasing mechanism removes expired entries and frees up space. The results of this thesis show that a small overhead is produced by the Jobs perform slightly slower than aspects, however the advantage of the jobs is that they execute after tasks.
Also [Kla14] extends XVSM. Semantic XVSM enables the use of semantic models and knowledge inside the XVSM space. The introduction of these Semantic Web technolog, offers powerful query capabilities for selecting communications items and their enrichment with implicit

[9] [KCH14] page 64.
[10] [Wat14] page 1.

information. The coordination logic of Semantic XVSM is platform independent, can use standardized languages, and manages the logic at run time by using standard operations of XVSM.
TuCSoN has been extended in [UOdlT+14] with security mechanisms for a P2P agent coordination framework used by heterogeneous health organisations to exchange Electronic Health Records of patients.The main aim is to secure the interactions within the P2P network so that different health organisations can freely search data in other health organisations without prior integration among them. TuCSoN has been extended with asymmetric key cryptography models in order to reach this goal. Future work will focus on defining policies that can detect if the emerging behaviour of the actors performing the queries is that expected within the system. Furthermore detailed logging features will be developed in order to track back all the access to documents and subscriptions to different types of events will be developed in the future.
Also KLAIM has been extended in [ADNL15]. RepliKlaim enriches Klaim with primitives for replica-aware coordination. The main aim is is to offer suitable solutions to the problems of data distribution and locality in large-scale high performance computing. RepliKlaim offers the possibility to *specify and coordinate the replication of shared data items and the desired consistency properties so to obtain better performances in large-scale high performance computing applications.* [11]
An interesting application scenario was presented in [BDL+14] and focuses on energy optimisation in the context of lifts management. Here a distributed optimisation strategy aims to reduce the electricity bill. In order to achieve this goal [BDL+14] uses LINC which is based on the ideas of Linda. The main challenge is to find the right balance between grid and stored energy. So fare this application scenario has only been simulated but there are plans to test this strategy on a real lift.
So one can see that space based computing systems are constantly improved in order to fit new application scenarios or the adress topics like security, mobilty, life cycle management, usability, extensibilty or scalability amongst others. Since the information on the space based computing systems is sometimes scattered it is important to understand that I might have simply not found a few space based computing systems. Therefore this thesis is probably a good start in order to get a good overview of what is out there but of course it's probably not an entirely complete picture.
Future work could focus on the quantitative classification process of the introduced space based computing systems but one would need to think about a framework that makes the spaces really comparable to each other.

[11] [ADNL15] page 14.

APPENDIX A

Glossary

Although not all space-based computing solutions were mentioned explicitly in this thesis, it still makes sense, to introduce all of them shortly to give the interested reader, a complete picture, including references and links. The space-based computing solutions are presented in a not too technical way in order to reduce complexity.
If any space-based computing solutions are missing, this is unintentional and simply due to the sheer volume of solutions available.

A.1 Apache River

In 2007, Sun's Jini contribution was accepted into the Apache incubator with the project name 'River' to improve visibility of the Jini development. The Apache River project website[1] gives a decent overview of the project including concepts, source code, a community area and much more. Since Jini is still the better known name in the community, further details can be found in the glossary section under Jini.

A.2 AspectKE*

AspectKE* [YMA+10] is a member of the KLAIM family and presents a programming language which offers an aspect extension to Klava [BDNP01]. The main focus of AspectKE* is to enforce security aspects in an untrusted environment.
The main characteristics of AspectKE* can be described as follows:

- *AspectKE* can express a large set of security policies, especially those based on future behavior of executing processes.*[2]

[1] `http://river.apache.org/doc/spec-index.html`
[2] [YMA+10] page 31.

- AspectKE* offers a high-level program analysis.
- AspectKE* combines load-time static analysis and runtime checking, which enhances the performance.
- AspectKE* can *express dynamic properties of an executing process in combination with static properties.*[3]

At the moment AspectKE* *can only monitor processes and command the processes to break or proceed from its advice*[4] so further research has to be undertaken to make it useable for a wider range of scenarios to expand the range of useability. Source code or project websites are currently not available.

A.3 AutoevoSpaces

IntaMission's AutevoSpaces had been presented in 2003 on the company's website, which no longer exists. AutoevoSpaces SSI by IntaMission was the first commercially available product to provide semantic consistency with Sun's JavaSpaces, ensuring the correctness and flexibility of large distributed mission-critical applications.
A free trial used to be offered by Sun but, once again, the provied link is no longer active. The main features of AutoevoSpaces are:

- Building a common data model to deliver improved collaboration and service reliability, while simplifying the underlying infrastructure.
- Allowing service providers to consolidate and migrate complex systems in a way that reduces risk and drives down the cost of change.
- Monitoring services end-to-end across the domain from a single point, allowing the business to drive down support costs while boosting the quality of service.
- Enabling grid computing through Autevo-processing power is spread across your entire network, maximizing the use of your existing hardware.

Since no source or literature is available on this topic, it's hard to say what the strength and weaknesses of IntaMission's AutevoSpaces are.

A.4 B-Linda

Linda with bound types - or in short B-Linda - extends the basic Linda model with the aim to supply Linda with a better tolerance to programming errors and unwanted interactions in open context. It was introduced in 2002 by Alain Gibaud and Philippe Thomin [GT02]. The

[3] [YMA+10] page 31.
[4] [YMA+10] page 31.

main difference to the basic Linda model is that tuples are characterized by a b-type and not only by their structures. B-types are defined by Gibaud and Thomin *as a triple (`Ste`, `Se`, `Sc`) in which `Ste` represents the structure of the tuple, Se its semantics and `Sc` its scope.*[5] It also introduces two new operators in addition to the classical operators `out`, `in`, `rd` and `eval`, namely the `definetype (st, se)` operator (allows the dynamic creation of a b-type) and the `undefined` operator (allows a process to remove itself from the scope of the `b-type bt`).
The model avoids false matching. First of all it allows dynamic partitioning of the tuple space and secondly, *the partitions built are homogenous because they contain information sharing the same structure and the same semantics, used in the same context.*[6] This aspect can be seen as the main advantage compared to Linda.
This space based computing solution is a solid option with a theoretical approach but it also exists as an implementation, presented and tested in C. Unfortunately the only literature concerning this topic is provided by Philipp Tomin and Alain Gibaud themselves and only two other works have cited their article.

A.5 BaLinda K

BaLinda K represents a further development of BaLinda Lisp and is presented in [YF96] by C.K. Yuen and M.D. Feng. The "K" stands for "hard C" because the language of the K version superficially resembles C. BaLinda K is especially useful *for the construction of I/O interfaces and execution control mechanisms and has potential as a tool for system program implementations.*[7] On top of this, any inheritance anomaly can be handled effectively with Balinda K and its active objects. The paper provided by C.K. Yuen presents a lot of practical examples but no source code for practical testing is made available.

A.6 BaLinda Lisp

BaLinda (Biddle and Linda) Lisp, is a parallel execution of Lisp which was first presented by C.K. Yuen and W.F. Wong in 1990 [YW90]. The language was created due to the following two requirements: Firstly, to execute parallel threads and secondly, to allow these parallel threads to communicate with each other, including task-blocking to wait for a condition to arise. BaLinda Lisp omits a lot of the classical Lisp features and it also only uses the `in`, `rd` and `out` operations of Linda, neglecting the `eval` operation.
In [YW90] Yuen and Feng present the fork-and-join and construct in more detail which marks two threads as parallel and shows where the two threads merge back into one. The paper also discusses the idea of speculative parallelism, first presented in [YFY93] in more detail. This idea is of relevance when it comes to topic of performance improvements. Theoretical material as well as practical implementations are provided by the authors. The authors have presented a lot

[5] [GT02] page 833.
[6] [GT02] page 834.
[7] [YP96] page 438.

of code examples, to help understand the principle of operation of this space based computing implementation. Unfortunately no source for personal testing is available.

A.7 Bauhaus Linda

Bauhaus Linda [CGH97]was developed by N. Carriero D. Gelernter and S. Hupfer with the aim to improve groupware systems. Bauhaus derived from Linda and offers a framework which facilitates the common Linda framework through the following measures:

- *elimination of the distinction between Linda tuples and tuple spaces. Instead Bauhaus Linda introduces the construct multiset (mset).*[8]
- *elimination of the distinction between Linda tuples and ani-tuples.*[9]
- *reduction of the distinction between passive data-objects and passive data-objects. The* `eval` *operation in Linda loses its relevance in Bauhaus Linda, where the* `out` *operation creates processes and passive data elements.*[10]

At the same time Bauhaus Linda is more powerful than Linda because it can handle more than one tuple space *but a hierarchy of "agent interaction spaces"*.[11]
In this paper the authors present a number of groupware implementations which allow a better understanding of how Bauhaus Linda can be used effectively. As a matter of fact, Bauhaus Linda *supports both synchronous and asynchronous interactions, both human and software agents and both messaging and shared memory communication.*[12]
The paper has been cited in many successive works and can be considered influential in the sector of space-based computing technologies. No source has been provided for personal testing.

A.8 BISSA

BISSA owes his name to a storage place where old Sri Lankan farmers used to store their harvest. This space-based computing implementation is relatively new and a good introduction was presented by [WWF+10] in 2010. On their project website the BISSA-team states that *BISSA is a scalable and distributed tuple space implementation which can be used as a decoupled , easy to program distributed shared memory abstraction for distributed applications.*[13] Bissa has a two level architecture which consists of the following:

- *A peer to peer scalable implementation of a tuple space which can be used as a java library.*[14]

[8] [CGH97] page 312.
[9] [CGH97] page 312.
[10] [CGH97] page 312.
[11] [CGH97] page 312.
[12] [CGH97] page 319.
[13] `http://bissa.sourceforge.net/about.html`
[14] `http://bissa.sourceforge.net/about.html`

- *An in-browser tuple space implementation which can be used as a tuple space for java script applications.*[15]

Both can either act as standalone implementations or collaborate with each other.[16] BISSA supports the standard operations `read`, `write` and `take` but also a `subscription mechanism` based on tuple templates.
The BISSA-Team presents a very complete picture of BISSA and the source is also available for testing. It seems, however, that not many have cited the idea of BISSA in their own work and so one has to question the relevance of BISSA. That being said, it is certainly a big step for the empowerment of web gadget communication.
Further information can be found under the following link.[17]

A.9 Blitz

Blitz[18] is an open source implementation implementing the JavaSpaces spec. The project is meant to provide a basis for JavaSpaces-compliant Jini-services and to be installed easily with an installation package. The key features of Blitz JavaSpaces (Pure Java Edition) 2.1.6 are described on the website[19] as follows:

- *Pure Java - no native libraries required*
- *No External Database Requirement - just configure and go*
- *JINI 2.1 support - with backward compatibility to JINI 2.0.x*
- *JavaSpace 05 support*

Besides implementing the classical JavaSpaces, Blitz allows to configure three different persistence profiles. They are currently[20]:

- *Persistent - Blitz behaves like a fully persistent JavaSpace.*
- *Transient - (de setting) causes Blitz to act like a disk-backed cache. No logging is performed and, when Blitz is restarted, all state (including Join state etc.) is lost.*
- *TimeBarrierPersistent - provides a performance versus persistence QoS*[21] *tradeoff. In this mode, changes made more than a certain number of milliseconds ago are guaranteed to be persistent. More recent changes are not guaranteed persistent but may be persistent. This mode provides the developer with a means of balancing persistence needs against performance.*

[15] `http://bissa.sourceforge.net/about.html`
[16] [WWF+10] page 1.
[17] `http://bissa.sourceforge.net/doclist.html.`
[18] `http://www.dancres.org/blitz`
[19] `http://www.dancres.org/blitz/blitz_js.html`
[20] `http://www.dancres.org/bjspj/docs/docs/install_guide.html`
[21] Quality of Service

A very useful characteristic of Blitz is that it offers various tools, including amongst others a dashboard, which provides graphic access statistics such as memory usage, instance counts and a number of active transactions.
Although the project website supports the reader with a lot of detailed information, no theoretical in detail papers are available.

A.10 Blossom

Blossom is a C++ version of Linda with extensions and is presented in [GSW97]. The main motivation for the introduction of Blossom was to make tuple spaces, as known in 1997, more efficient and safer. In short Blossom tries to improve the program performance and the security, *in the sense that the likelihood of programming errors is reduced.*[22] Blossom suggests four enhancements to the classical Linda model in order to reach this goal:

- Blossom uses strongly typed tuple spaces meaning that *each tuple space is created with a specification that gives the number, order and type of each field of the tuples legally allowed.*[23]
- Blossom uses field access patterns, where tuple fields can only be retrieved by specifying a value, or allowing a wild card variable.
- Blossom also applies access patterns to tuple spaces.
- Blossom allows users to look into tuple spaces, which is referred to as tuple space assertions.

[GSW97] shows that the proposed enhancements help to detect or eliminate parallel programming errors, however there are no performance tests, which indicate how much the user can gain through these measures. In the paper the authors also state that they are hoping to add additional performance enhancements, but no literature has been found on whether or not they have realized their plans.
[GSW97] is cited in other works, but there are no further appearances of Blossom itself, neither in publications nor on the web.

A.11 Bonita

Bonita is described in [Row97], [RW97] and proposes new primitives - Bonita primitives - that use the same template matching and tuple space concept as the Linda primitives but provide asynchronous access to tuple spaces, which avoids long waiting times for primitives that are blocked most of the time.
The following primitives are introduced:

[22] [GSW97] page 2.
[23] [GSW97] page 6.

- `rqid = dispatch (ts, tuple | template, destructive | nondestructive)`

  The `rqid` primitive can be seen as a copy-collect and collect functionality. It returns the requestidentifier `rqid` that can be subsequently used with other Bonita primitives to count the number of copied or moved tuples.

- `arrived (rqid)`

  `arrived (rqid)` is a non-blocking primitive which can return the values `t` true or `f` false to show if the tuple has arrived or not. An invalid `rqid` returns the value false.

- `obtain (rqid)`

  `obtain(rquid)` is a blocking primitive which waits until the tuple or result specified by the `rqid` argument is available.

[Row97] shows that BONITA has a better performance than C Linda because BONITA *pipelines access to the tuple space*[24] which means a performance gain in the majority of times. This also leads to the conclusion that BONITA is *better suited to the type of co-ordination required in agent systems.*[25]
[Row97] and [RW97] are often cited in literature and have shown that performance can be improved using simple measures. No source for testing has been found on the internet.

A.12 C-Linda

The C-Linda [Gel85] programming language is the combination of the C and the Linda languages. C-Linda provides the tools and the environment necessary to combine distinct processes into a complete parallel program.
The parallel operations in C-Linda are orthogonal to C, providing complementary capabilities necessary to parallel programs. C-Linda programs make full use of standard C for computation and other non-parallel tasks; C-Linda enables these sequential operations to be divided among the available processors. Since C-Linda is implemented as a precompiler, C-Linda programs are essentially independent of the particular (native) C compiler used for final compilation and linking.[26]
C-Linda, along with Fortran, can be seen as the oldest of all Linda-like implementations and a lot of articles about it have been published. Nevertheless, in 2012 other programming languages have evolved that support the aims of Linda even better.

A.13 C++ Linda

In [DF96a] R. Drucker and A. Frank present C++ Linda which combines the ideas of Linda tuple spaces and the C++ object orientated programming language. *The idea is to take a C++ program*

[24] [RW97b] page 3.
[25] [RW97b] page 4.
[26] [Sci05] page 10.

and transparently replace its data storage mechanism with that of the tuple space. Distributed C++ variables are automatically stored as unique tuples in the tuple space rather than being allocated storage in internal memory (on a stack or in a heap).[27]
Also [Gal97] offers a solution in C++ Linda, with the aim to handle large tuples efficiently.
One can see that more research on this topic has been carried out already, with the last bachelor of science thesis having been published by [Slu07] in 2007. In this work we get a very detailed overview of the possibilities of C++ Linda, and the source is also available under `http://sourceforge.net/projects/cpplinda/files/cpplinda/`.
By embedding Linda and its primitives in C++ language, one obtains a new, parallel version of the C++ language. An object-oriented language like C++ definitely has its advantages compared to C. The object-oriented aspects in C++ enable aspects such as inheritance, encapsulation and reuse. However, one has to bear in mind that the use of object-oriented language can also have disadvantages.

A.14 Comet

Comet [LP05] is a scalable peer-to-peer content-based coordination space, which has been implemented on the JXTA project[28] with the aim to support *wide-area P2P environments, which require rich data expressibility, flexible matching, and scalable performance.*[29]
Comet offers a global virtual shared-space that can be associatively accessed by all system peers. Such peer groups provide a secure environment where only member peers can access the services available in the respective tuple space. Peers can belong to several spaces and an application can dynamically switch between coordination services associated with these spaces.
Comet operates on mainly two phases:

- Bootstrap phase: peer nodes try to join other peers and in doing so they construct a routing table of available peer nodes.
- In the Running phase 2 modes are executed:
  - Stabilization mode: peer nodes respond to queries issued by other peers.
  - User mode: peer nodes provide coordination services.

Comet has been cited quite often in 2005 when it was first presented, but it seems that not many new developments have been added to the original system since. A comet cloud is presented under `http://nsfcac.rutgers.edu/content/comet-cloud`, but it can't be confirmed how many of the ideas in Comet have been used in it.

[27] [DF96] page 31.
[28] `http://www.jxta.org`.
[29] [LP05] page 1.

A.15 Corso

Corso[30] extends Linda through a virtual shared memory space of Java objects and has been presented in 2001 by the tecco Software Entwicklung AG, a spin-off of the Technical University of Vienna at the institute for computer languages.
Corso offers the following features [Ang03]:

- Every object has a single identifier.
- The caching behavior can be defined by the developer.
- Transactional security is provided.
- Various recovery scenarios are possible.
- Notifications.
- Automatic garbage collection.
- Authorization mechanisms.
- Support of C++, Java and .Net.
- Process control allows components to start processes which allows a parallel execution.

Corso has had a great deal of attention from both industry and academia with numerous articles published. An overview can be found under. `http://www.complang.tuwien.ac.at/eva/researchpublications.html`
Corso represents the predecessor of XVSM[31] which is a development of the Space Based Computing Group at the Technical University of Vienna and will be described in more detail under appendix point 103.

A.16 Crudlet

Crudlet[32] is an event based architecture for connecting XML based interface structures to Jini service layers and was introduced in 2001.
In 2001 Crudlet, which also followed the concept of tuple spaces, was praised for its purity of separation of powers between the front end design layer (XSL/XML), the business logic layer (JavaBeans), the backend messaging layer (JavaSpace and agents) and database layer (JDBC). In 2016 the Crudlet project site is no longer available and although the Java package org.crudlet. can still be found on some websites, not much information is available on it. Some information can be found on the following two websites:

[30] Coordinated Shared Objects
[31] eXtensible Virtual Shared Memory
[32] creation, retrieval, updating, deletion, scheduling of life cycle activity, existence checking, and generic templating of various objects in the context of business rules

- `http://www.infolets.com/1006361585/`
- `http://javaspaces.homestead.com/files/javaspaces.html`

A.17 D-Tuples

DTuples [JXJY06] are a Linda like peer-to-peer tuple space middleware built on top of distributed hash tables with the aim to simplify the development of distributed cooperation and coordination tasks. The higher layer abstraction supports this aim
In the application level, the DTuples can be used instead of the publish/subscribe model and messagepassing model.[33]
The tuples in the DTuples are stored in distributed hash table based peer-to-peer tuple storage and support the following operations:

- `in ()`
- `rd ()`
- `out ()`
- `copy-collect ()`: In Linda the multiple `rd ()` is a problem, but the *copy-collect(ts1,ts2,template) primitive copies all available tuples that match the given template from one specified tuple space (ts1) to another specified tuple space (ts2).*[34]

DTuples is an efficient way of improving peer-to-peer networks, however apart from this paper no other works or sources could be found.

A.18 DepSpace

DEPSPACE [BACF08] is implemented as a secure, fault-tolerant and intrusion-tolerant tuple space in Java. *The main objective of the system is to provide an extended tuple space abstraction that can be used to implement Byzantine fault tolerant applications.*[35] DEPSPACE is covering the following topics:

- Replication
- Confidentiality offers three protection types for tuple fields:
  - public: the field is not encrypted
  - comparable: encrypted field but comparisons are possible through a stored hash
  - private: the field is encrypted

[33] [JXJY06] page 1.
[34] [JXJY06] page 1.
[35] `http://www.navigators.di.fc.ul.pt/software/depspace/`

- Access Control
- Policy Enforcement

The source can be found under `http://www.navigators.di.fc.ul.pt/software/depspace/` and there seems to be ongoing research towards this topic, although [BACF08] is still the only paper available at the moment.

A.19 dRuby and Rinda

dRuby and Rinda were presented in [Sek09] as the distributed object environment and shared tuplespace implementation for the Ruby language and are included as part of Ruby's standard library.
dRuby is one of several RMI libraries for Ruby and its aim is to extend Ruby method calls to other processes and other machines. It might be important to know that dRuby is *exclusively for Ruby and written purely in Ruby.*[36]
Rinda implements tuple spaces in Ruby and allows multiple Ruby processes to easily share data. This is accomplished by using the Ring server which allows the following operations:

- `read`
- `read_all`
- `write`
- `take`
- `notify`

Although Rinda has a lot of advantages like its simple parallelizable programming model for Ruby it's only well suited for non-critical tasks because it doesn't scale very well and it is not persistent, meaning that if it crashes while there are tuples in the space they are all lost.
dRuby and Rinda have quite a big community, there are a lot of blogs on the web with advice and hands-on information concerning this topic. But only a few useful theoretical papers exist besides [Sek09], e.g. [JYM08] and [TS90]. A lot of information can be found on `http://www.druby.org`.

A.20 EgoSpaces

EgoSpaces [JR06] is a coordination model and middleware for ad hoc mobile environments.
EgoSpaces's fundamental concept is the view abstraction. A view is a collection of relevant data (or context) items that are defined by meta data, the software agents that own the data, the hosts on which those agents are located, and the network paths that connect the application in

[36] [Sek09] page 3.

question to the hosts holding the data items. The key novel aspect of the view abstraction is its provision of asymmetric coordination.[37]
[JR06] also presents three use cases, namely a subscription music service, an agent registration and migration service and a collaborative puzzle game in detail.
The EgoSpaces prototype implementation and further useful documentation are available on `http://www.ece.utexas.edu/~julien/egospaces.html`

A.21 Eiffel Linda

Eiffel Linda [Jel90] combines the programming language Eiffel and the Linda model. In Eiffel Linda tuples are defined as objects and the class tuples implement all the operations we know from basic Linda (`out`, `in`, `rd`, `eval`).
The advantage of treating tuples as objects lies in the fact that *tuples can be assigned to variables, passed to procedures and stored inside other tuples.*[38]
Eiffel Linda has been cited quite often in other works, but apart from this paper no further work on Eiffel Linda could be found.

A.22 eLinda

eLinda, as described in [Wel05] and [WCC01], is an extended version of Linda in Java. The main aim is *to simplify the development of distributed applications and to enhance the efficiency of communication in a distributed memory environment.*[39]
Three major key extensions have been introduced in eLinda:

- A *Programmable Matching Engine*[40] allows efficient searching algorithms.
- Two output operations are offered, a `point-to-point` operation and a `broadcast` operation, which use replicated data.
- eLinda supports Multimedia data.

The work of George Wells seems to be generally well cited and eLinda introduces some intersting features which were missing in Linda. Unfortunately no source could be found on the internet.

A.23 Encrypted Shared Data Spaces

Encrypted shared data spaces [RDD+08] use encryption schemes, RSA[41] and discrete logarithms, that ensure confidentiality of the data space content in open, possibly hostile, environments.

[37] `http://users.ece.utexas.edu/~julien/egospaces.html`
[38] [Jel90] page 79.
[39] [WCC01] page 1.
[40] PME
[41] Rivest, Shamir and Adleman

The big asset of an encrypted shared data space is that it supports tuple matching even over the encrypted data space. The data space does not need to decrypt tuples to perform the search which keeps real content safe from *nosey hosts.*[42]
The key management in encrypted shared data spaces is quite easy because agents do not have to share keys for reading and writing tuples. The key management is handled by a fully trusted server which is responsible for all the key-related operations.
The encrypted shared data space itself is used for storing and retrieving plaintext, partially encrypted or completely encrypted tuples. In addition it also performs encrypted searching operations, authenticates valid clients, and safely stores encryption and decryption keys.
The encrypted shared data space offers a different approach to other security mechanisms in space-based computing implementations because it does not focus on access control alone. No source code or project website is available.

A.24 Entangled

Entangled is a distributed hash table (DHT) based on Kademlia[43], as well as a peer-to-peer tuple space implementation with the aim to create peer-to-peer network applications that require synchronization and event handling.
Entangled is written in Phyton and makes use of the following operations:

- `publish`
- `search`
- `remove`

In addition a very simple user interface is available. The source code can be downloaded from `http://sourceforge.net/projects/entangled/files/`

A.25 Erlinda

Erlinda is a parallel computing framework for Erlang. Information as well as the source can be found on the *Erlinda project website.*[44] *Erlinda supports a number of abstractions for creating parallel applications such as:*[45]

- Tuple Space model based on Linda memory model (JavaSpaces)
- MPI like APIs for communication (Scatter/Gather)
- Map/Reduce
- Master/Slave (Computing Farm)

[42] [RDD+08] page 15.
[43] `http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html`
[44] `http://code.google.com/p/erlinda/`
[45] `http://erlang.wikidot.com/cn\protect\kern+.2222em\relaxerlinda#toc3`

- Service oriented (OTP)
- Clustering
- Agents and Mobililty
- Messaging oriented (Queues/Topics)
- Service/Resource discovery mechanism (similar to JINI)
- Integration with other languages and middlewares
- Code Server (similar to JINI/RMI)
- Security

No detailed description is available but the source code is documented well, therefore testing shouldn't be too difficult.

A.26 Fly Object Space

Fly[46] is a lightweight object space that can distribute and coordinate information on clusters of computers in the form of objects.
One advantage of Fly is the generalized object server which means that it supports Java, Ruby and Scala for space clients. Fly implements the same core operations (write,read,take) as JavaSpaces, but doesn't implement garbage collection which leads to an increase in performance.
The Fly Object website can be found under `http://www.flyobjectspace.com/` and supports the reader with the source code, documentation and videos. Fly offers a commercial as well as a free version.

A.27 Forth-Linda

A detailed description of Forth-Linda was presented in 1991 under `http://www.ultratechnology.com/4thlinda.html`. Forth-Linda extends the Forth Language. Forth-Linda was intended to support homogeneous environments with a single network protocol.
Forth-Linda implements the following five operations:

- `eval`
- `rd`
- `out`
- `rm`: read and remove tuple from description
- `rq`: request an active tuple to execute

[46] `http://www.flyobjectspace.com/`

Apart from the website[47] no other Forth-Linda references could be found, implicating that in 2012 Forth-Linda is not considered state-of-the-art anymore.

A.28 Gaia

GAIA [RHCN02] introduces Active Spaces as immersive computing environments for context-aware applications. *It provides services for location, context, events and repositories with information about the active space.*[48]
The main application area of GAIA can be seen in the teaching or in the work environment, and it serves as a platform to explore the various aspects of ubiquitous computing.
User data and applications are abstracted into a user virtual space and can be mapped dynamically to the resources located in the current environment.[49] Therefore, users can move from one active space to another, seamlessly integrating into new spaces.
GAIA functions best in small networked environments where the available resources in the space can be centrally managed by a kernel.
A fair number of articles have been published on this topic and the work of Román has been cited often in literature.

A.29 Geo-Linda

Geo-Linda presented in [PCBB07], addresses the problem of *detecting different kind of movement patterns of devices.*[50] In other words it addresses use cases like the insertion of a product in a shopping cart, the loading of containers in trucks or boats or helping visually impaired people to take the bus.
In order to support such use cases Geo-Linda has introduced the following innovations:

- Geometric addressing helps to precisely detect moving patterns.
- A `readOnce (s,p)` operation is introduced, *which returns a new tuple that matches the pattern* `p` *and whose shape intersects the addressing shape* `s`.[51]
- A `lostOne (s,p)` operation is introduced, *which returns a tuple that matches the pattern* `p`. *The shape of the returned tuple intersects the addressing shape* `s` *and has disappeared from the visible tuple space.*[52] This operation is only viable when the tuple has used the `readOnce (s,p)` operation before.

Implementations have shown that Geo-Linda can support such use cases but besides [PCBB07] no information precisely dealing with Geo-Linda could be found. Never-the-less [PCBB07] has been cited quite often.

[47] `http://www.ultratechnology.com/4thlinda.html`
[48] [RHCN02] page 4.
[49] [RHCN02] page 15.
[50] [PCBB07] page 3.
[51] [PCBB07] page 4.
[52] [PCBB07] page 4.

A.30 GigaSpaces

GigaSpaces[53] is a commercial implementation of JavaSpaces with the goal to support high scalable applications and was presented in 1999. It provides tightly coupled caching, parallel processing, database and applications access, and Java Messaging Service[54] support. The infrastructure can store objects in distributed in-memory caches and supports clustering, replication, failover, load balancing and security.
GigaSpaces also offers a free Lite version that can be downloaded under
`http://www.gigaspaces.com/LatestProductVersion` with a subset of features. In 2012 it offers a wide set of services and is a professional space-based computing implementation.

A.31 GLinda

GLinda [KA07], developed in Prolog, is a local desktop grid environment extended with the Linda model and was developed with the aim to support communication that offers associative content-based messaging between hosts without having information about their physical location.
Communication and synchronization is carried out through the GLinda server using the coordination model of Linda. The Linda operations are implemented as Prolog database operations. *Further, in Prolog variables are untyped so GLinda tuples are untyped, avoiding multiple problems in tuple description and matching.*[55]
All threads within the same process share a common tuple space which is why the changes made on the tuple space by a thread are immediately visible to all the threads. Typical GLinda commands are:

- `glup`: upload
- `glbcast`: broadcast
- `glrun`: execution
- `glclean`: delete

GLinda also leaves room for improvement such as the availability of multiple tuple spaces. Unfortunately there is no source for testing on hand and this is the only paper available.

A.32 Globe

GLOBE [LS02] is a distributed and replicated tuple space that provides high availability and high scalability for large-scale Internet based systems. The level of availability and scalability

[53] `http://www.gigaspaces.com/`
[54] JMS
[55] [KA07] page 2.

of the tuple space is adjustable through the number of defined replicas and partitions in the system.
[LS02] has been cited often in literature but no source for testing is available.

A.33 Grinda

Grinda[56] has been presented amongst others in [CM08b] and in [CM08a]. It is a tuple space implementation for the *Globus Toolkit 4*[57] and can be used to coordinate distributed tasks without knowing host identities and network topology.
Grinda has two modules[58]:

- A client-side module that allows client applications to access Grinda operations.
- A server-side module that is deployed on the Globus Toolkit and represents a Grinda service.

Clients use the service to store and retrieve tuples. The services are connected to one another and can communicate to distribute or search tuples. The distribution is either based on past experience or on a master slave strategy. The source code is available under `http://grinda.sourceforge.net/doc/index.html` and the last updates have been presented in 2007.

A.34 Gruple

Gruple[59] extends Groovy in order to provide a simple abstraction for coordination and synchronization of threads and processes. Information about Groovy and its source can be found under `http://gruple.codehaus.org/TuplespaceUsage` Gruple supports the following operations:

- `put` - insert a tuple into the space
- `get` - read a tuple from the space (non-destructively)
- `take` - take a tuple from the space (a destructive read)

Further development for Gruple is needed to be truly useful, e.g. transactions or distribution.

A.35 GSpace

GSpace [RCVS04] is a distributed shared data space that supports a variety of distribution policies. With this idea a few requirements have to be fulfilled:

[56] Grid + Linda
[57] `http://www.globus.org/toolkit/`
[58] `http://grinda.sourceforge.net/doc/index.html`
[59] Groovy + tuple

- Tuple distribution requirements are declared separately from the application code.
- A set of different distribution policies are embedded that can be used to tailor the behavior of different applications.
- New distribution policies can be added easily.

[RCVS04] also measured the performances of several distribution policies with the result that every application usage pattern requires a different distribution policy.
GSpace differs from other systems because its flexibility is extendable[60].
Whilst a few papers are available on this topic, there is no source code for this interesting approach.

A.36 Helios Tuple Space

The Helios Tuple Space library [CM08b] enhances the Linda model through additional library calls in the C Language, e.g. program termination, jobs can be split in portions and be given to free workers or mutual exclusion.
These measures add more flexibility to the Helios Tuple Space Library, also good performance has been proven by tests presented in [CM08b].
No source code is available.

A.37 Heterocera

Heterocera is a *sinatra*[61] based associative memory system by David ten Have with all information uploaded on `https://github.com/dave5/heterocera-server` in 2011 or 2012.
Heterocera is an implementation of an associative memory system and is inspired by Linda with the aim to *simplify and decouple communications between a range of systems of varying capability.*[62] Therefore the memory space is seen as a web server and address locations are *URLs*[63]. No additional information about Heterocera apart from this website could be found.

A.38 HTML Page Spaces

A reference about HTML Page Spaces was found on `http://c2.com/cgi/wiki?TupleSpace` but the link provided is no longer active and no other references could be found. Therefore no closer description is possible.

[60] [RCVS04] page 12.
[61] `http://www.sinatrarb.com/`
[62] `https://github.com/dave5/heterocera-server`
[63] Uniform Resource Locator

A.39 Info Spaces

InfoSpaces [BMSV] addresses the problem of the interaction between ubiquitous devices by using tuple spaces at user level. The following simple drag-and-drop operations are supported:

- `copy out`
- `move out`
- `copy in`
- `move in`

Possible use cases include information kiosk applications, control applications and wireless data collection applications. JXTA is used to implement InfoSpaces and a few papers have been presented but the source is not available.

A.40 Jada

Jada [CR97] has been developed on the basis of the PageSpace project [CR02] and is a combination of Linda and allows Java applications and respectively threads to enter a shared object space using operations like

- `read`
- `in`
- `out`

Applications can access as many object spaces as they like and each object space can be either local or remote (local object spaces can be shared between threads; remote tuple spaces can be shared between java application or applets).[64]
Jada was a solid solution back in 1996 combining the ideas of tuple spaces and Java. The source code is available under `http://www.cs.unibo.it/~rossi/jada/`.

A.41 JavaSpaces

JavaSpaces is a distributed shared memory that provides a high-level means of creating collaborative and distributed applications. It is a core Jini service but can be used alone as well. JavaSpaces orientates itself to Linda with the difference being that JavaSpaces runs on a *Java Virtual Machine*[65] and therefore can operate easily on many different platforms. Tuples in JavaSpaces are Java objects and JavaSpaces provides the following operations on them:

- `write`: puts a new entry in the space.

[64] `http://www.cs.unibo.it/~rossi/jada/`
[65] JVM

- `read`: returns a copy of an entry matching a particular template.
- `take`: removes an entry matching a particular template from the space and returns it to the caller.
- `notify`: notifies the caller whenever entries that match the template are added to the space.
- `snapshot`: minimizes the serialization that occurs whenever entries or templates are used.

Since its release in 1998, JavaSpaces has seen many ups-and-downs as well as a set of quite similar solutions. Much is known about JavaSpaces and therefore it's still an interesting solution.

A.42 Jedi

JEDI[66] was presented in [CDNF01] and was developed with the aim to support the development and operation of event-based systems. JEDI supports the following operations:

- `open`: opens the connection with the event dispatcher.
- `close`: closes the connection with the event dispatcher.
- `subscribe`: subscribes the object.
- `unsubscribe`: unsubscribes the object.
- `dispatch`: allows the object to generate an event notification.
- `getEvent`: retrieves the event address from the object.
- `hasEvents`: checks if any of the subscribed events has events.
- `moveOut`: disconnects temporarily from the event dispatcher.
- `moveIn`: reconnects to the event dispatcher after a moveOut.
- `move`: moves objects to other location.

[CDNF01] also gives an implementation example, the OPSS workflow management system[67] and shows how much such a system can benefit from JEDI.
The advantages of JEDI lie in the easy re-configurability of the system, the easy distribution/replication of components and the easy plug & play of components.
A few papers dealing with JEDI are available but no source code can be found.

[66] Java event-based distributed infrastructure
[67] WFMS

A.43 Jini

Jini[68] is a service orientated network architecture for the construction of distributed systems. It was introduced in 1998 by Sun and was transferred to Apache[69] under the project name River in 2007. The purpose of Jini is to allow groups of services and users to participate into a single, dynamic distributed system that offers:

- Simple access
- Easy administration
- Easy sharing mechanisms
- Spontaneous interactions
- Self-healing mechanisms
- Security
- Code mobility

The main operations in Jini are:

- `discovery`: find a lookup service.
- `join`: register your service with a lookup service.
- `lookup`: find a service in the lookup service.
- `invoke`: use the local object to call the service.

Because Jini is implemented in Java, many applications require a Java virtual machine to be present. In 2016 information as well as source code can be found under `http://river.apache.org/`.

A.44 JION

JION[70] [BLG12] is a middleware based on JavaSpaces that supports the development and the communication in D-MANETs[71] through the centralization of all services.
JION supports:

- Event logging
- Tuple matching

[68] Jini = the devil in Swahili
[69] `http://river.apache.org/`
[70] JavaSpaces Implemetation for Opportunistic Networks
[71] Disconnected mobile ad hoc networks

- The classiscal JavaSpaces operations

JION is distributed under the terms of the GNU General Public License and downloads as well as further information can be found on the project website under `http://www-irisa.univ-ubs.fr/Abdulkader.Benchi/JION.html`.

A.45 Joyce Linda

Joyce Linda [McD92], [PM91] is derived from the Joyce programming language [Han02] and the Linda model. *The traditional use of Linda's eval primitive is subsumed by Joyce/Linda's concurrent instantiation of agents, though agent termination does not result in the creation of a data tuple unless explicitly coded.*[72]
Joyce Linda uses capabilities to provide secure inter-process communication between agents having a common ancestor. Every agent possesses two capabilities with unique values, called self and caller.
All required concurrency is implicitly defined in the semantics of Joyce/Linda and its compilers need not infer concurrency or vectorize statements. The granularity of parallelism is the agent, not the statement.[73]
Joyce Linda was used by McDonald [McD92] to add practical examples to courses which were meant to teach concurrent programming. Joyce Linda has been mentioned often in literature but apart from the papers provided by McDonald and Pinakis no other sources are available.

A.46 JParadise

JParadise^{TM}[74] is an implementation of the Linda model by the Scientific Computing Associates, Inc., specifically designed for distributed computing environments. JParadise^{TM} supports

- the communication between completely different applications running at potentially different times.
- persistent tuple spaces.
- multi tuple spaces.

JParadise^{TM} and Linda can be used together which leads to applications that are both parallel and distributed. JParadise^{TM} can be used for financial and risk analysis, computational chemistry and biology, seismic analysis, EDA and fluid dynamics.
A lot of information can be found under `http://www.lindaspaces.com/products/paradise_overview.html`

[72] `http://ccnuma.anu.edu.au/cap/cap/reports/report94/Joyce.html`
[73] `http://ccnuma.anu.edu.au/cap/cap/reports/report94/Joyce.html`
[74] `http://www.lindaspaces.com/products/paradise.html`

A.47 JXTA Spaces

JXTA[75] was introduced by Sun as an open source protocol in 2001 with the aim to support peer-to-peer communication [Li01]. Napster[76] and Gnutella [77] did use this concept of communication.
Although JXTA is not based on the idea of tuple spaces its still an approach that needs to be presented to better understand space-based computing implementations.
JXTA is based upon a set of open XML protocols and implementations are available for Java SE, C/C++, C# and Java ME. JXTA peers create a virtual overlay network which allows a peer to interact with other peers through firewalls and NATs[78]. Each peer has a constant identification number and can be identified even if it changes location.
JXTA offers two types of peers:

- edge peers: are transient and have a low bandwidth network connectivity.
- super-peers: are divided into:
  - rendezvous peers: coordinates the peers in the JXTA network.
  - relay peers: helps peers behind firewalls or NATs to take part in the JXTA network.

Peers in JXTA can locate each other and have communications with each other through Pipes. JXTA is not bound to any platform. JXTA has received a lot of attention in the past but in November 2010, Oracle announced its withdrawal from the JXTA projects and since 2011 no information about the future of JXTA has been given. `http://jxta.kenai.com/` offers still information about JXTA.

A.48 Kernel Linda

Kernel Linda is an extension of Linda and *designed to support programming at the system level*[79] with the goal to be hardware independent. Kernel Linda is used for different use-cases, *e.g. target tracking, pattern recognition and speech processing.*[80]
[Haz93] and [Lel90] give a short introduction to Kernel Linda. The main characteristics of Kernel Linda are that it supports

- the communication between operating processes and user processes,
- the implementation of multiple tuple spaces,
- a mixed language environment, because it provides a set of language-independent data-types and

[75] Juxtapose
[76] `http://www.napster.com/`
[77] `http://www.gnutellaforums.com/`
[78] Network Address Translation
[79] [Haz93] page 16.
[80] [Haz93] page 17.

- asynchronous communication.

In comparison to the classical Linda, Kernel Linda doesn't make use of the eval function but still makes use of an environment-based process model.

A.49 Klava

Klava[81] [BDNP01] is the implementation of KLAIM [DNFP98] in Java *and it is used for implementing distributed applications that can exploit mobile code and run over a heterogeneous network environment.*[82]
Klava supports the following operations:

- `in`
- `out`
- `read`
- `eval`: creates a new process on a local or remote node
- `newloc`:creates a new remote node

Klava has been first presented in 1998 and since then it has been enhanced continuously, e.g. with improved security features [BDN02]. The project website can be found under `http://klava.sourceforge.net/` and includes all available papers and the link to the source code `http://klava.git.sourceforge.net`.

A.50 L^{2}imbo

The L^{2}imbo, as presented in [DFWB98a] and in [DFWB98b] extends the classical Linda model. The following extensions have been made to improve multicast and undirected communications:

- Bonita primitives allow the clients to access tuple spaces asynchronously.
- Multiple tuple spaces support performance, partitioning and scale issues.
- System agents offer a variation of different services.
- Deadline based operations are supported.
- Tuple typing is supported.

The website given in [DFWB98a] didn't work anymore and no further information on the web could be found.

[81] KLAIM in Java
[82] `http://klava.sourceforge.net/`

A.51 Lacios

LACIOS [83] [ZBS09] extends Linda and is a data-oriented coordination language which focuses on the design and implementation of multi agent systems used for transportation applications.
In LACIOS agents can publish or update their status and even condition their interaction with the environment which is of great use in transportation applications.
[ZBH10] enhances LACIOS with security features that make sure objects are not fraudulent and only agents with access rights to the object can execute operations on them.
LACIOS is a good solution for transportation applications and use cases have been presented in [ZBS09] but no source code is available. Since LACIOS is a relatively new approach it is likely that more enhancements are going to be developed.

A.52 Lana

Lana [BR02] is based on Java and the Linda tuple space model *that includes concepts for communication, mobility, security and connection recovery in order to support autonomy.*[84]
Lana supports autonomy through asynchronous method calling, protection domains, an associative message board for communication by value and an event model that allows programs to securely delegate the handling of events to others.[85]
Events are deposited into the tuple space and retrieved immediately, later or never by the corresponding agent.
Lana has been of interest when it comes to security and it follows, like many others, a capability-based security policy.
No project website or newer papers are available.

A.53 Law-Governed Infrasturcture

Law-Governed Infrastructure[86] [MMU01] is a coordination model that is related with Law-Governed Linda without changing the Linda model at all. It allows agents and processes to communicate with each other under a certain policy, which is called the law of the group. Policies exist for security issues as well as for coordination issues.
LGI characteristics are:

- Laws under LGI are sensitive to the content of the tuples being handled.
- Laws are sensitive to the state of agents.
- Enforcement of laws can occur at either the client, the server, or anywhere in the network.
- LGI supports different levels of security.

[83] Language for Agent Contextual Interaction in Open Systems
[84] [BR02] page 1.
[85] [BR02] page 26.
[86] LGI

Every agent can join an arbitrary number of groups and follows their different law as necessary. A law in LGI can be described as a reactive rule which executes an operation on a certain event. Like in LGL, a controller monitors everything to secure that policies are fulfilled.
At the moment no cryptographic techniques are used to enhance security which can be seen as a drawback.
LGI has found approval in the scientific community but like with LGL not enough new research results are available.

A.54 Law-Governed Linda

Law-Governed Linda[87] [ML95] extends the Linda model with law-governed architectures in order to enhance the security of tuple spaces. To reach this goal Law-Governed Linda forces all processes to adopt a set of specific rules which are monitored by a controller who has a copy of the law that applies to the specific tuple/tuple space. oOnly if a process fulfills all the laws it can execute operations on the tuple/tuple space.
Law-Gowerned Linda has been cited often in the literature but no source code or project website are available.

A.55 LighTS

LighTS [PB05] is a Java-based implementation of the Linda tuple space and was originally developed as the core of LIME. The main idea was to develop a very light core, which hardly wastes any resources, and still allows to easily extend *features like persistency, security, or remote access.*[88]
LighTS also *offers an adaptation layer that has the same interface of the tuple space implementation provided in LighTS but allows for loading different implementations of the adapters at startup, each converting the operations provided by the LighTS interface into those of other tuple space implementations.*[89]
LighTS is quite flexible and easy to extend and a lot of suitable information is provided in literature such as in [BCP05] as well as on the project website. Also the source code is available for testing.

A.56 Ligia

Ligia [MW98] extends the Linda tuple space and is a Java-based implementation which includes garbage collection of tuple spaces and agents.
The garbage collection mechanism in Ligia collects unnecessary tuple-spaces using a graph, where the reference information is maintained. However, tuples within the tuple space cannot be garbage collected selectively. But taking into account that Ligia was presented 1998 it is

[87] LGL
[88] `http://lights.sourceforge.net/`
[89] `http://lights.sourceforge.net/`

a very good implementation and shows how garbage collection can positively contribute to a space-based computing implementation.
Only this paper was found on the internet but the work was cited in quite a few articles and is certainly relevant.

A.57 Limbo

Limbo extends the Linda model and is described in [DWFB97]. The aim of Limbo is to *provide a better support for adaptive mobile applications.*[90]
The following extensions have been made to the Linda model:

- Multiple tuple spaces can be used to enhance consistency, security or performance.
- An explicit tuple type hierarchy is introduced.
- *Tuples with explicit QoS*[91] *attributes are introduced.*[92]
- A set of system agents are introduced that provide services like monitoring, the creation of new tuple spaces and the propagation of tuples between the spaces.

This is the only article that can be found concerning Limbo and also no sources are found on the internet. Therefore one has to come to the conclusion that Limbo can simplify the implementation of various mobile applications but does not present the state of the art.

A.58 LIME

LIME[93] a Java-based middleware adopts ideas of the Linda model and converts them to a mobile environment. It was first presented in [PMR99] and further publications followed, including amongst others in [PMR00], [MPR01] and [MPR06].
The main three measures to adapt the Linda model for mobile requirements are:

- transiently shared tuple spaces – *where each agent has access to an interface tuple space that is permanently associated with that agent and transferred along with it when movement occurs.*[94] The Its[95] can be shared with other agents which creates a transient tuple space because *its content changes according to the migration of agents.*[96]
- the tuple location has to be controlled continously.

[90] [DWFB97] page 5.
[91] Quality of Service
[92] [DWFB97] page 5.
[93] Linda in a Mobile Environment
[94] [PMR99] page 3.
[95] Interface tuple space
[96] [PMR99] page 3.

- the reaction to events has to happen as quickly as possible since the situation in a mobile environment changes all the time. The run-time support, often a stationary agent , *continuously monitors the underlying layers for system events.*[97]

LIME has been refined since 1999 and also led other programmers to further enhancements, e.g. a SecureLIME [HR03]. Also TinyLIME and TeenyLIME have been created out of the LIME middleware.
A source, detailed documentation and other information is available on the project website. [98]

A.59 LIME II

The LIME II middleware is presented in [AAHC09] and can be seen as a complete reengineering of the classical LIME package. The aim of LIME II is to support use cases in *dynamic environments where distributed transactions are not practical and unannounced disconnections are frequent events.*[99]
The implementation presented can be seen as a MANET[100] and was realized with commercial pocket PCs. The main strength of this solution is that the transportation to far away nodes does not cause a big challenge, because the message is sent from one space to another with a simple routing algorithm.
[AAHC09] also presents a power analysis in their paper which shows that LIME II *can run continuously for about 128 minutes on a fully charged battery. .*[101]
LIME II is relatively new and shows that the idea of LIME has a lot of relevance for the mobile challenges. So far the source code is not available and no further extensions to LIME II have been presented.

A.60 Limone

Limone [FRH04] is a lightweight coordination middleware for wireless ad hoc networks that *consists of logically mobile agents and physically mobile hosts.*[102] The main concepts of Limone are:

- Limone presents an asymmetric coordination style where agents only communicate with agents that satisfy their self-set acquaintance policy. In the case this acquaintance policy is fulfilled the agents remember these agents by storing them on a so called acquaintance list.

- *Limone eliminates remote blocking and complex group operations.*[103]

[97] [PMR99] page 7.
[98] `http://lime.sourceforge.net/Lime/papers.html`
[99] [AAHC09] page 53.
[100] Mobile Ad Hoc Network
[101] [AAHC09] page 58.
[102] `http://mobilab.cse.wustl.edu/projects/limone/`
[103] `http://mobilab.cse.wustl.edu/projects/limone/`

- Limone provides timeouts for all distributed operations and reactions.

This very high grade of security increases the communication under the agents and this leads amongst others to fast response times in a changing environment. The source of Limone can be downloaded under `http://mobilab.cse.wustl.edu/projects/limone/`. Limone is an interesting approach which has also been developed further in [DNG10].

A.61 Linda

Linda[104] [Gel85] is a model for communication and co-ordination of parallel processes and can be seen as the mother of all space-based computing implementations.
The main concept in Linda is that of a tuple space, an abstraction via which processes can communicate. *A tuple space can be seen as a bag of tuples, a tuple being an ordered sequence of heterogeneously typed objects. Linda also supports templates which are different from tuples because a field of a template may be typed, but have no value.*[105]
The main operations, for manipulating tuple spaces, supported by Linda are:

- `out (t)`: adds tuple `t` to the tuple space. Non-blocking.
- `in (t)`: either, removes any tuple `t` that matches template `s` and, assigns values of actuals in `t` to variables in `s` or, blocks.
- `Rd (t)`:like `in (t)` but the tuple is not removed from the tuple space.
- `Eval (t)`: like `out (t)`, but `t` must be evaluated.

Linda was originally implemented in C and Fortran, but due to its simplicity it had found a wide community which added a big number of extensions in various programming languages to the original. But Linda doesn't only have supporters but has also received criticism [ZEN92].
Today the original Linda model is not prepared to deal with all the demands developers and industry make from a space-based computing implementation but it has still provided - and will provide - the basis for many new successors.
A good and complete user manual of the commercial version of Linda is provided by the Scientific Computing Associates, Inc. the under `http://www.lindaspaces.com/downloads/lindamanual.pdf`.

A.62 Lindacap

Lindacap [UDI09], [UWJ07] extends Linda with a capability-based control mechanism to provide a more refined control for open distributed systems without losing the flexibility of Linda.[106]
Capabilities can provide all sorts of information on objects, including which methods can be executed on the objects.

[104] Linda is named for Linda Lovelace who was an actress in the porn movie Deep Throat
[105] [DWR95] page 126.
[106] [UDI09] page 105.

The basic input/output primitives in Lindacap extend the basic primitives of Linda with multicapabilities as additional parameters.[107]
The additional information, provided by the capabilities given to the agents, leads to better control and more efficiency in distributed systems.
Lindacap has been cited often in literature, especially in the context of garbage collection [UDI09]. No source code is available.

A.63 Linearizable Byzantine Tuple Space

Linearizable byzantine fault-tolerant tuple space [108] [NBCdSFCL07], [Bes06] offers a solution for ad hoc networks and mobile agents where simple operations use simple quorum-based protocols and more complicated operations use consensus-based protocols.
In LBTS write (out) and read (rdp) operations can be implemented using quorum-based protocols and consensus is needed for read-remove (inp) operations.
The overall system model is based on a set of servers from which less than a fourth may be faulty and on an unlimited number of client processes, from which arbitrarily many can also be faulty.[109] This is extremely useful when it comes to non-trusted processes in dynamic distributed systems.
LBTS offers the following advantages:

- It is wait-free.
- It is faultolerant.
- It is linearizable.
- It is reliable and available.

[NBCdSFCL07] and [Bes06] have been cited quite often in literature but no source code or project website is available.

A.64 LinqSpace

Linqspace [Gel11] extends the XVSM[110] middleware solution with the .NET environment LINQ[111] with the aim to enrich XVSM *with uniform and versatile query capabilities.*[112]
LinqSpace is still a prototype which has a final architecture, core functionality (read, write, take), a relational database for storage and [Gel11] offers considerations how to improve LinqSpace in the future.

[107] [UWJ07] page 16
[108] LBTS
[109] [NBCdSFCL07] page 11.
[110] eXtensible Virtual Shared Memory
[111] Language Integrated Query
[112] [Gel11] page 3.

A.65 Linux Tuples

LinuxTuples[113] is an open-source implementation by Will Ware that extends the concept of tuple spaces, designed to run on a networked cluster of Linuxmachines. LinuxTuples support the following operations:

- `PUT`
- `GET`
- `READ`
- `GET_NB`: *A non-blocking version of* `GET`.[114]
- `READ_NB`: *a non-blocking version of* `READ`.[115]
- `DUMP`: creates a list of the tuples currently in the space.
- `LOG`: creates a logfile of the tuple server.

More information about LinuxTuples and the source code are available under `http://linuxtuples.sourceforge.net/` and `http://sourceforge.net/projects/linuxtuples/`. The source code was last updated in 2009.

A.66 LuaTs

LuaTS is a reactive event-driven tuple space, and was first presented in [LR03] and implemented in Lua[116]. As many other implementations it extends the original Linda model with the aim to improve the use in wide area network-based applications, where most implementations struggle due to their synchronous behavior when it comes to accessing the tuple space.
LuaTS only provides asynchronous calls, offers *a strong mechanism for retrieving tuples, supports code mobility and includes a reactive layer through which the programmer can adapt the behavior of the basic system calls* [117] Besides that LuaTS offers a reasonable access control.
Only [LR03] has been presented so far and only little information can be found on the Lua website[118].

[113] `http://linuxtuples.sourceforge.net/`
[114] `http://linuxtuples.sourceforge.net/`
[115] `http://linuxtuples.sourceforge.net/`
[116] `http://www.lua.org/`
[117] [LR03] page 731.
[118] `http://www.lua.org/`

A.67 LuCe

LuCe [DOLA99], [DOT00] and [DO01], combines the ideas of Java, Prolog and tuple centers in order to support Internet-based multi-agent systems.
The main difference of LuCe is that it makes use of tuple centers. *What makes a tuple center different from a tuple space is the notion of behavior specification, which defines how a tuple center reacts to an incoming/ outgoing communication event.*[119]
LuCe agents can interact by exchanging logic tuples through these tuple centers and can use the same operations as known from Linda (`out`, `in`, `rd`, `inp`, `rdp`).
LuCe is presented in a set of papers but no source code is available.

A.68 MARS/Moon

MARS [CLZ00] is a coordination architecture based on Linda and implemented in Java with the aim to enable agents to coordinate with other entities. The tuples are Java objects, and agents can retrieve or store them with the following operations:

- `read`
- `take`
- `write`

Agents can also associate reactions to the operations made on the space. Topics like security or garbage collection are not addressed at the moment.
MARS offers a graphical interface with which one can write tuples into the local tuple space, install and de-install reaction and launch agents.[120] [CLZ00] as well as the source code are available under `http://www.agentgroup.unimo.it/MOON/MARS/index.html`.

A.69 Melinda

Melinda[121] can be seen as the first implementation provided by Yale that addresses the problem of multiple tuple spaces and that really offers a distributed environment [Hup90].
The key idea of Melinda is that it treats processes as well as tuples as first class objects which allows Melinda to adapt much better to aspects like mobility or failure resistance. The communication procedures are very much the same as in Linda and therefore of no new scientific importance.
Apart from [Hup90] no one explicitly works with Melinda although multiple tuple spaces are an important part of research since its technical report which has been cited often in the academic community.

[119] [DOT00] page 2.
[120] `http://www.agentgroup.unimo.it/MOON/MARS/index.html`
[121] Linda with multiple tuple spaces

A.70 MobiS

MobiS [Mas99] is an extended version of PoliS [Cia91], which is a specification language based on multiple tuple spaces. MobiS offers a hierarchical structured system which can be described as a tree where spaces are presented as nodes and change over the course of time. Spaces are first class entities, can move and contain three types of tuples:

- ordinary tuples present sequences of values
- program tuples can be seen as agents
- space tuples are containing subspaces.

Communication is handled by putting tuples that represent messages in the same space. Mobility is modelled by the consumption and production of space tuples by rules [NIS10].
However, MobiS does not provide an explicit primitive for locations. It uses spaces both to model components and locations.[122]
MobiS definitely has its place in literature and has been cited often but it seems as if the planned extensions, e.g. the development of security features for the model, have not been implemented so far.

A.71 Network Spaces

NetWorkSpaces™ is a product of the Scientific Computing Associates, Inc. and was created to use clusters from within scripting languages like Matlab, Python, and R. NetWorkSpaces™ offers amongst others the following features:

- *Uses script-compatible globally shared workspaces to develop and run distributed/parallel script applications.*
- *Enables script processes to share data across networks.*
- *Network script processes can run either uncoupled or coordinated.*[123]

NetWorkSpaces™ is available as open source code as in a professional version which has to be licensed. A lot of information, e.g. user manuals, source code, can be found under `http://www.lindaspaces.com/products/NWS_overview.html`.

[122] [NIS10] page 42.
[123] `http://www.lindaspaces.com/products/NWS_overview.html`

A.72 Open Spaces

A.72.1 Open Spaces by Giga Spaces

OpenSpaces[124] is an open source initiative from GigaSpaces[125], designed to enable scaling out of stateful applications in a simple way using Spring. It is built around GigaSpaces' eXtreme Application Platform (XAP).
The forum is a way for GigaSpaces to stay in contact with the community, the website `http://www.openspaces.org` offers documentation, a wiki, the source code, extensions made by other developers et cetera.

A.72.2 Open Spaces

Another implementation which is also called OpenSpaces is described in [DHN00]. It is an object-oriented framework that supports static configurability as well as dynamic configurability of policies with the aim to introduce a space-based computing solution for various use cases.
There is no source code available for further testing and no other papers concerning OpenSpaces could be found.

A.73 Open Wings

Openwings[126]was developed by General Dynamics Decision Systems (formerly Motorola IISG) and Sun in 1999 as a set of open systems specifications for a framework that enables the development of highly available, secure, distributed systems for mission critical applications where systems are likely to come and go in an ad-hoc fashion. Openwings is an abstraction on top of various service discovery mechanisms, including Jini and provides a component framework for Service-Oriented Programming[127]:

- *Container services addresses the need for process lifecycle management, clustering/load balancing, security, handling of mobile code and overall service availability in a distributed environment.*[128]
- *Component services provide the mechanism to publish services as well as discover and use services that have been created and published by others.*[129]
- Connector services support component to component communication.
- Install services

[124] `http://www.openspaces.org`
[125] `http://www.gigaspaces.com/`
[126] `http://www.openwings.org`
[127] SOP
[128] `http://www.openwings.org/openwings-11/tutorial/Trail_Introduction/04_Container_Services.html`
[129] `http://www.openwings.org/openwings-11/tutorial/Trail_Introduction/03_Component_Services.html`

- Context services
- Management services
- Security services

The website `http://www.openwings.org` offers a detailed description on OpenWings including source code and tutorials. Most documents have been last updated in 2003. Since January 2013 the website is no longer available.

A.74 P-Linda

PLinda[130] [AS91] is an extension of Linda with the aim to create a model powerful enough for persistent data storage in addition to distributing computing without losing the *simplicity, expressiveness, and universality*[131] of Linda.
Persistent Linda extends Linda through:

- the qualities of transactions , *e.g. serialize-ability, crash recovery, and persistence.*[132]
- extended tuple patterns and relational database join semantics.
- in-place updates
- *hints (pragmas) to be supplied by the programmer so that data organization and process scheduling is more efficient.*[133]

Most of the classical Linda operations were refined and new operations were added and are explained in the user manual[134] on the project website[135]. The source code as well as articles are also available under `http://www.cs.nyu.edu/~binli/plinda`.

A.75 P4 Linda

P4-Linda [BLL93]consists of two implementations of the Linda programming model and the p4 parallel programming system[136], using both shared-memory and distributed-memory models for the underlying hardware.
One implementation is strictly a shared-memory implementation and uses monitors as the synchronization primitive whereas the other implementation is based on message-passing.[137]

[130] Persistent Linda
[131] [AS91] page 16.
[132] [AS91] page 3.
[133] [AS91] page 3.
[134] `http://www.cs.nyu.edu/~binli/plinda/manual.ps`
[135] `http://www.cs.nyu.edu/~binli/plinda/`
[136] no longer supported
[137] `http://heather.cs.ucdavis.edu/~matloff/LLinda/P4-Linda/tmp/p4-linda/README`

Both implementations were prototypes in and have not been developed any further. Information and program code can be still found under `http://heather.cs.ucdavis.edu/~matloff/LLinda/P4-Linda/NotesP4-Linda.html`

A.76 PadSpace

A.76.1 PadSpace

PadSpace [LT09] uses an XML space based on the ideas of the Linda coordination model with the aim to support the cooperation of Web applications, Web services and end users' local resources. In a PadSpace, users can directly manipulate pads, visual components, and can create new components.
Pad Space enables end users to interactively register their resources (mostly web-resources) into the XML-tuples pace, *and to interactively use those shared resources as visual components in combination with their own local visual functional resources through the automatic matching between function providers and function requesters.*[138]

A.76.2 PadlogSpace and PadlogSpace

PadlogSpace [LT10] is a further development of PadSpace and extends the service registration function and the service matching function of PadSpace with the goal to provide a semantic rule-definition mechanism for the service composition development.

A.77 PoliS

PoliS[139] is a coordination language written in Prolog that extends Linda with multiple tuple spaces with the aim to offer controlling coordination of distributed systems, for space and time computations [Cia91], [Cia94]. PoliS works with trees of nested spaces which can evolve dynamically in time. A space can contain:

- other spaces
- ordinary tuples: a structured data object that is a sequence of values
- program tuples: contain coordination rules to manage local activities

both other spaces and tuples of two types: ordinary
Agents perform Linda-like operations on tuples, spaces and places with the latter being a multiset of tuples.
It is a very well described implementation that shows a solid approach and has gained much respect in the academic field. Papers are available but no project website could be found.

[138] [LT09] page 8.
[139] Poli Spaces

A.78 Prolog-D-Linda v2

Prolog-D[140]-Linda v2 [Sut93] is the last version of five embeddings of the Linda model into Prolog. The first version was called muProlog [SP89] and allowed a single processor to run all the application processes and the centralized tuple space.
Prolog-D-Linda v2 supports applications running over an internet of processors with a distributed tuple space. *This implementation of Prolog-D-Linda runs on SICStus Prolog.*[141] Prolog-D-Linda v2 supports the `eval` operator, has a distributed tuple space, and provides i/o facilities for all processes in the system.
`http://www.cs.miami.edu/~geoff/ResearchProjects/PDL/` provides a download of Prolog-D-Linda v2 and several papers that describe predecessor versions.

A.79 PyBrenda

ByBrenda extends the Linda model, using Python. It is quite similar to C-Linda and uses also the same operations as such.
The source can be downloaded under `http://wiki.python.org/moin/PyBrenda` but the links to the project website are no longer active.

A.80 PyLinda

PyLinda extends the Linda model through

- multiple tuple spaces
- garbage collection
- sane non-blocking primitives
- bulk tuple operations

The former project website `http://www-users.cs.york.ac.uk/~aw/pylinda/` does not exist anymore.

A.81 Ruple

Ruple, an internet shared memory space, has been introduced by Rogue Wave Software[142] in [Tho02]. *Ruple is a distributed computing technology built on an Internet tuple space for XML documents, providing a loosely coupled asynchronous and anonymous link between multiple senders and receivers.*[143] The main difference to other space-based implementation is that Ruple

[140] distributed
[141] `http://www.cs.miami.edu/~geoff/ResearchProjects/PDL/`
[142] `http://www.roguewave.com/`
[143] `http://xml.coverpages.org/ni2002-03-01-a.html`

uses a document-centric approach *where XML documents are stored on the space and retrieved using XML query syntax (XQL).*[144]
Ruple can write robust, loosely coupled, collaborative multiway applications over the Internet.[145]
Ruple had quite a lot of attention back in 2002 but now all the weblinks on the Rogue Wave Software website are dead and only a few articles can be found on the internet.

A.82 Semantic Tuple Spaces

Semantic Web Spaces [TN04] is *a Linda-based coordination platform which allows the representation and management of Semantic Web information by extending the classical Linda model*[146]:

- The tuplespace holds data and knowledge
- *The classical operations are defined to operate upon the tuples as data*[147]
- Operations upon agent scope, namely - addscope, rmscope are added
- To add or remove tuples as knowledge operations - claim, retract are added
- To query tuples as knowledge semantic matching relations through the operation- rdiftrue are added

In 2008 [NSKMR08] presented a survey showing the state of the art focusing on that topic, which shows that semantic web spaces are a very interesting topic. Even though there is no source code available there are, quite a few papers which show the development over the years quite well.

A.83 SecOS

Secure Object Spaces[148] [VBO03], [BDN02] extends Linda with fine-grained access control based on locking in order to strengthen the security of space-based computing systems without being too restrictive. A lock can be seen as a specific value that represents the key to a given tuple. SecOS offers a set of different variants for the locking process:

- Symmetric key locking: One key locks and unlocks the tuple.
- Asymmetric key locking: A public key locks the tuple and a private key can unlock it.
- Fine grained access control at fields and tuple level.

[144] [KLF04] page 2.
[145] [Tho02] page 12.
[146] `http://www.ag-nbi.de/research/semanticwebspaces/index.html`
[147] `http://www.ag-nbi.de/research/semanticwebspaces/index.html`
[148] SecOS

Ideas of SecOS have influenced SecSpaces [BGLZ02]. More research regarding cryptographic protection for objects exchanged over the Internet needs be done to enhance SecOS for this use case.
No project website or source code is available.

A.84 SecSpaces

A.84.1 SecSpaces

SecSpaces [BGLZ02], which is quite similar to SecOS, is a Linda-like coordination model that supports secure data-driven coordination in open environments.
SecSpaces introduces two new fields which can both add value to the coordination model:

- Partitioning fields: The tuple space partitioning is achieved through the introduction of a partition field in the tuples. An agent can only enter the field if he knows the name of the partition. This partitioning mechanism avoids all agents having the same view on the data contained in a tuple space.
- Cryptographic fields: The cryptographic fields can distinguish between agents who can only execute read operations and those who can only execute take operations through the use of asymmetric cryptography.

SecSpaces offer a good approach to secure tuple spaces and tuple fields and therefore they are quite popular in the scientific community, but no source code is provided.

A.84.2 WSSecSpaces

WSSecSpaces extends the coordination primitives of SecSpaces in order to use Web services.
WSSecSpaces supports a secure coordination, the integration of Linda with Web technologies and the loose coordination of Web Services. The security of data transmitted on the channel is preserved by HTTPS transfer protocols used in the SOAP service invocation protocol.
More research, e.g. on other cryptographic solutions or the quality of service, can be done to improve WSSecSpaces. No source code is available.

A.85 SemiSpace

SemiSpace is a light weight open source interpretation of tuple spaces based on ideas from JavaSpaces[149] with clustering built using Terracotta[150].
SemiSpace was said to be non-intrusive, easy to configure and integrate. Unfortunately the SemiSpace website is down and so no further information about SemiSpace can be given.

[149] `http://www.theserverside.com/news/thread.tss?thread_id=55069`
[150] `http://terracotta.org/`

A.86 SmallSpaces

SmallSpaces are open source implementations of the Linda/Tuplespace programming model. A reference of SmallSpaces can be found under ,http://en.wikipedia.org/wiki/Tuple_space but the link provided there is no longer active and no other references could be found.

A.87 SQLSpaces

SQLSpaces were developed by the Collide Research Group at the University of Duisburg-Essen, Germany and a wide range of information on it can be found under the project website http://sqlspaces.collide.info/ which also includes the SQLSpaces downloads.
SQL[151]Spaces combine the idea of the tuple spaces concept with a relational database that can be seen as the backbone of the implementation but can hardly be noticed by the programmer because of the features, e.g. notifications, expiration, versioning and extended query mechanisms, which are offered by the tuple spaces. SQL supports a lot of different programming languages. In order to use SQLSpaces efficiently either MySQL[152] or the free Java database system HSQLDB[153] needs to be installed on your system.
On the 23th of August 2012 the latest version of SQLSpaces was published with JSON as the new message format. However, XML is also still supported.
Furthermore, a web workbench is presented that has a live view on the spaces where tuples can be changed, created et cetera. The SQLSpaces already have a rich user group in academia as well as in enterprise, mainly because it offers continuous improvement to make SQLSpaces more flexible.
A lot of papers have been presented concerning this topic including [WGH07].

A.88 Swarm Linda

SwarmLinda bases on the ideas of swarm intelligence implemented in a Linda-based system. The concept of SwarmLinda is presented in [GMT08], [CMT04], [TM03] and in [MT03]. SwarmLinda takes three major principles from swarm systems in nature and transfers them to feasible algorithms in the Linda world:

- Simplicity: SwarmLinda only uses a simple set of rules, which hence leads to a small resource usage.
- Dynamism: SwarmLinda reacts to a dynamic changing environment and does not expect that things stay static.

[151]Structured Query Language
[152] http://www.mysql.com
[153]http://hsqldb.org/

- Locality: Entities in SwarmLinda only observe their neighbor entities and *make decisions based on their local view.*[154]

Following this rules SwarmLinda achieves desired characteristics such as scalability, adaptiveness and fault-tolerance.
[GMT08] has made performance tests with SwarmLinda and shown that *SwarmLinda demonstrates self-organized characteristics while allowing an easy visualization tool of the behavior of SwarmLinda.*[155]
There is no project website and concepts like LIME seem to have found a bigger acceptation with programmers: That being said, SwarmLinda still offers a good approach to bringing Linda to a mobile environment.

A.89 Tagged sets

Tagged sets [OH05]are a virtual shared memory approach *that relies on tags based on propositional logic to lock and select values*[156] with the aim to support scenarios like shared data repositories, message passing or publish/subscribe algorithms.
Each element in the virtual shared memory is a value with an associated tag, and values are read or removed from the virtual shared memory by matching the tag.
Tags are implemented as propositional logic formulae, and selection as logical implication, so the resulting system is quite powerful.[157]
Tagged sets provide good security because tags are usually associated with a key that defines the protection grade and only if a user presents his key he is allowed to execute an operation on the tag.
Apart from [OH05] no other works related to tagged sets could be found.

A.90 T-Spaces

TSpaces [WMLF98], [LMW99] and [LCX$^+$01] have been introduced by IBM in 1998. TSpaces is a combination of tuple spaces, databases and Java with the aim to *generate a platform independent repository that connects heterogeneous systems in to one common computing platform.*[158]
The tuple operators as well as the matching algorithms are enhanced compared to the classic Linda model. The database layer offers additional stability and durability whereas Java brings portability to this implementation.
Advantage of TSpaces are that it offers

- asynchronous messaging
- database functionality

[154] [TM03] page 4.
[155] [GMT08] page 2716.
[156] [OH05] page 15.
[157] [OH05] page 1.
[158] [LMW99] page 1-2.

- XML support
- all kinds of different use cases

Negative aspects are that TSpaces are incompatible with lower versions and the lack of security provided.
TSpaces is a probably one of the most thought through tuple space implementation for Java out there. It can be downloaded under `http://www.almaden.ibm.com/cs/TSpaces/`. On this site one can also find a lot of information and documentation. Apart from the download an Enterprise Suite is also available.
The last update in the paper section `http://www.almaden.ibm.com/cs/TSpaces/papers.html` was made in 2003.

A.91 TCP Linda

TCP Linda is a product of the Scientific Computing Associates and it is based on Linda and TCP[159].
TCP's Linda architecture is modular and its semantics are the same as in Linda and an content-addressable tuple space, which makes it easier to build applications. One well known use case of TCP Linda is Gaussian 03[160].
TCP Linda can be seen as the earliest commercial version and one of the most widespread implementations of a virtual shared memory for supercomputers and clustered systems.
Information and free trials for TCP Linda for Linux and Mac OS X can be found under `http://lindaspaces.com/downloads/evaluation.html`.

A.92 TeenyLIME

TeenyLIME is a model and middleware for Wireless Sensor Networks and has so far been presented in [CMMP06], [CMMP07], [CMP+09] and [CCD+11]. The aim of TeenyLIME is to *support applications where sensing and acting devices themselves drive the network behavior*[161] without relying on an external base station.
TeenyLIME is an extension of LIME and therefore the coordination operations are basically the same. The big difference to LIME is that the TeenyLIME tuple spaces are physically located on each and every device and that the communication is only granted with one-hop neighbors. In addition each device has a different set of one-hop neighbors and therefore each device has a different view of the tuple space. This set-up can display a variety of useful coordination scenarios, when it comes to collaborative tasks.
Interesting use cases of TeenyLIME are described in [CMP+09] and [CCD+11], which shows that TeenyLIME can be best used in real-world Wireless Sensor Networks, such as the monitoring of heritage buildings or the adaptive lighting in road tunnels. The source code and a wiki for further questions are available under `http://teenylime.sourceforge.net/`

[159] Transmission Control Protocol
[160] computational chemistry program, `http://www.gaussian.com`
[161] [CMMP06] page 1.

A.93 TIBCO ActiveSpaces

TIBCO ActiveSpaces is a distributed peer-to-peer in-memory data grid that enables heterogeneous applications to share, exchange, and process data in real time.
This approach means the capacity of the space scales automatically as nodes join and leave. Replication assures fault tolerance from node failure as the space autonomously re-replicates and redistributes lost data.[162]
Programming APIs are available in Java and C. TIBCO offers a designated blog which focuses on the development of ActiveSpaces[163] which is up to date and one can see that efforts are made to constantly improve TIBCO ActiveSpaces.
Since this is a commercial solution good product information is available.

A.94 The KLAIM family

KLAIM[164] is a space-based computing implementation *designed to program distributed systems consisting of several mobile components that interact through multiple distributed tuple spaces.*[165]
KLAIM has been presented in many papers, amongst others in [DNFP98], [BdNPF98], [DNL00] and [BBDN$^+$03].
The development of KLAIM has been mainly influenced by process calculus (π-calculus) and the Linda coordination model with the aim to improve issues like scalability and modularity. Over the time various KLAIM implementations have been presented:

- cKLAIM[166] can be regarded as a variant of the π-calculus, which can be characterized by features such as process mobility and asynchronous communication via so-called localities (instead of channel-based communication). In simple words, KLAIM allows agents to move to the location of the tuple space they want to communicate with.

  KLAIM can

  - create processes
  - create variables
  - create localities
  - create nodes
  - move processes between nodes

  For those who want to read about the syntax and the semantic of KLAIM in more detail, further information can be found in [BBDN$^+$03].

[162] `http://www.tibco.com/products/soa/in-memory-computing/activespaces-enterprise-edition/default.jsp`
[163] `https://www.tibcommunity.com/blogs/activespaces`
[164] Kernel Language for Agents Interaction and Mobility
[165] [BBDN$^+$03] page 88.
[166] coreKLAIM

- μKLAIM[167] extends cKLAIM with tuples and pattern-matching.
- OPENKLAIM is an extension which is designed to *enable users to give more realistic accounts of open systems.*[168] This is reached through *equipping KLAIM with mechanisms to dynamically update allocation environments and to handle node connectivity.*[169]
- HOTKLAIM[170]
- METAKLAIM
- O'KLAIM[171] adds object-oriented features to KLAIM
- X-KLAIM[172] *exends KLAIM with a high level syntax for processes: it provides variable declarations, enriched assignments, conditionals, sequential and iterative process composition*[173] X-KLAIM is based on the Java middleware KLAVA [BDNP01].

KLAIM is quite popular and there are a lot of papers concerning this topic and many extensions to cKLAIM are presented.

A.95 TinyLIME

TinyLIME is an extension of Lime and has been presented in [CGG+05c], [CGG+05b] as well as on the TinyLIME website[174]. The operational setting in which TinyLIME operates *assumes that sensors are distributed sparsely throughout a region, and need not be able to communicate with one another. The monitoring application is deployed on a set of mobile hosts, interconnected through ad hoc wireless links. Some hosts are only clients, without direct access to sensors and others are equipped with a sensor base station, which however enables access only to sensors within one hop, therefore naturally providing a contextual view of the sensor subsystem.*[175] This operational setting can be found useful in disaster recovery areas and military settings.
Technically *TinyLIME is implemented as a layer on top of LIME with specialized components deployed on sensors and base stations.*[176] TinyLIME works with a transiently shared tuple space, which stores tuples containing the sensed data. Another change is that TinyLIME works with motes, which are distributed loosely in the environment and communicate only with the base station, when the base station is in their reach. These motes are only shown by TinyLime if they are connected to a base station. For accessing sensor data TinyLIME agents make use of

[167] Micro-Klaim
[168] [BBDN+03] page 100.
[169] [BBDN+03] page 100.
[170] Higher-Order Typed KLAIM
[171] Object Orientated KLAIM
[172] eXtended KLAIM
[173] [BBDN+03] page 135.
[174] `http://lime.sourceforge.net/tinyLime/index.html`
[175] [CGG+05a] page 5.
[176] [CGG+05a] page 26.

the same operations as proposed by LIME, but only read operations are available (rd, rdp and rdg).
Reactions also work as in LIME, with two extensions:

- It can be decided *how frequently the data received through a reaction can be refreshed.*[177]
- Conditions are accepted by the reactions, e.g. to react only between a certain temperature range.

TinyLIME is very well described both on the website as in papers. A source code of this open source solution is also available on the website.

A.96 Triple Space Communication

TripCom[178], a research project with international partners from academia and industry and funded by the European Commission, represents a new form of network-based communication which aims to provide a platform which allows coordinated access to distributed semantic data. TripCom combines technologies like tuple spaces, web services and the semantic web with the aim to:

- allow many different clients to interact with each other, giving each client the possibility to publish data with different semantics into the space.
- allow the usage of different semantic data formats (such as the Resource Description Framework[179]).
- *allow coordinated access to data through an extended Linda coordination model.*[180]
- provide a scalable communication infrastructure for (Semantic) Web services.
- allow easy data distribution and access control.
- improve fault tolerance.

The goal of the project is to provide an infrastructure for the web of machines based on the principles of the web for humans.[181]
The source and a great deal of information can be found on the project website under `http://www.tripcom.org`. TripCom is possibly one of the most innovative approaches of all space based computing implementations which will need just a little more developement in order to become a standard.

[177] [CGG+05b] page 5.
[178] Triple Space Communication
[179] RDF
[180] `http://www.tripcom.org/faq.php#tripcom4`
[181] `http://www.tripcom.org/faq.php`

A.97 TuCSoN

TuCSoN[182] [OZ98a], [RO02], [NVCO10], [ORRV03] is a hybrid coordination model which aims to *support the design and development of information-oriented applications based on mobile agents.*[183]
TuCSon differs from Linda because it focuses on tuple centres (instead of tuple spaces), which allow to define and tailor the communication channel to the respective use case. In easy words, tuple centers are *programmable tuple spaces.*[184] The language to programme the tuple spaces is called ReSpecT. Tuple centers act in the same way as the common tuple space but their behavior can be enriched so as to encapsulate the coordination rules.
Those tuple centres are spread on network nodes (Internet nodes) and identified by their IP (or logic) addresses. A node can host an arbitrary number of tuple centres, called coordination context.
To access a TuCSoN coordination context, agents must negotiate their entrance after the concept of ACC[185].
TuCSon supports Java and tuProlog, and can be downloaded under `http://alice.unibo.it/xwiki/bin/view/TuCSoN/Download`. TuCSoN is a very complete solution which has been under current development since 1998 and has been tested in all kinds of use cases and a fair number of papers have focused on it. The project website is `http://alice.unibo.it/xwiki/bin/view/TuCSoN/`.

A.98 UML Spaces

UML-Spaces implement tuple spaces using UML[186] and have been introduced out of the motivation that UML is one of the most commonly used notations for software systems.
In order to implement UML-Spaces only a few adaptions within UML have been necessary according to [AR01]. For the programmer it might be useful to know that the operations read, take and write can only be used synchronously. Additionally the newly introduced «use» associates processes with the tuple spaces they use.
[AR01] shows in his paper that already a good amount of research has been conducted in this area, e.g. [LKG+99]. Nevertheless there are still a lot of topics which seem worth researching, for example *bringing UML-Spaces to distributed systems with logical and physical mobility.*[187]

A.99 VLOS

VLOS[188] is presented in [MCW02] and [CM03] and can be described as a distributed operating system based on tuple spaces.

[182] Tuple Centres Spread over the Network
[183] [OZ98a] page 3.
[184] `http://www.lia.deis.unibo.it/phd/materials/courses/`
[185] Agent Coordination Context
[186] Unified Modelling Language
[187] [AR01] page 134.
[188] Virtually Linda Operating System

In VLOS tuple spaces are created without anything associated with it, except some basic types and capabilities.
Security in VLOS is granted through a capability-based approach. Capabilities in VLOS normally consist of a unique identifier, the object type and the name of the tuple space with which the object is associated as well as of a set of rights, a cryptographic and a hash function [UWJ07].
In order to even better customize tuple field matching expressions [CM03] introduces the MiniMe matching expression language. MiniMe allows the specification of new field matching expressions but is still in its infancy.
VLOS has received some attention from the scientific community but no research results have been presented recently.

A.100 Xcoordination Application Space and Xcoordination Coordination Space

A.100.1 Xcoordination Application Space

The Xcoordination Application Space, also called AppSpace, is a .Net based implementation by Xcoordination[189] which aims to make writing asynchronous and distributed applications easier by combining a set of existing technologies:

- CCR[190]: handles asynchronous communication by defining asynchronous components, called workers, and make them as easily usable as synchronous components.
- WCF[191]: optional possibility to handle communication
- XMPP[192]: optional possibility to handle communication

The concept of workers introduced in application space is crucial since the space is just made up of workers which asynchronously communicate via a channel:

- Asynchronous workers
- Synchronous workers
- Local workers
- Distributed workers

Features like the puplish/subscribe pattern, asynchronous timed actions, basic security features, different transport services et cetera are supported by Xcoordination Application Space.
The Xcoordination Application Space is available as a free as well as a commercial version. The free version of AppSpace V1.3 can be downloaded under `http://xcoappspace.codeplex.com/`.

[189] `http://www.xcoordination.org/home`
[190] Microsoft Concurrency Coordination Runtime
[191] Windows Communication Foundation
[192] Extensible Messaging and Presence Protocol

The Xcoordination Application Space is a very well described, up-to-date and under current development with the aim to continuously improve it and add new features.

A.100.2 Xcoordination Coordination Space

The Xcoordination Coordination Space[193] can be seen as a complementation to the Xcoordination Application Space because it minimizes the traffic between workers in the space.
Communication and synchronization is carried out by simply writing, reading and taking shared, structured, synchronized data, as if they were collections of local data.[194]

A.101 XMIDDLE

XMIDDLE [MCE01], [MCZE02] is a middleware for mobile computing and allows applications the transparent sharing of XML documents with other hosts without any fixed network infrastructure lying underneath. XMIDDLE allows the on-line and off-line access to data and provides all hosts with consistent data.
XMIDDLE manages its data through trees that are described by means of XML documents and it supports the following primitives:

- `link`: This primitive links a tree to a remote host. *When linked and connected to a remote client host, the server host records the name of the client host, the branch it is linking to, and the linking point in the LinkedBy table.*[195]
- `unlink`: The unlink primitive changes the local LinkedFrom table and unlinks the not needed part of the tree branch.
- `connect`: The connect primitive first searches for remote hosts and their branches they are linked with and then connects them so that the required data can be provided.
- `disconnect`: The disconnect primitive allows a hosts to work off-line

XMIDDLE is a solid and well elaborated space-based computing implementation where papers, documentation as well as the source code are available on the project website `http://xmiddle.sourceforge.net/`.

A.102 XML Spaces

XML[196]Spaces [TG01b], [TG01a] extends the Linda model with the aim to support web-based applications.
It uses XML documents in addition to ordinary tuple fields in order to coordinate entities with the Linda-primitives. The following enhancements have been made to improve the Linda model:

[193] `http://www.xcoordination.org/coordination_space`
[194] `http://www.xcoordination.org/coordination_space`
[195] [MCZE02] page 94.
[196] Extensible Markup Language

- Tuple fields can carry XML documents.
- Multiple matching relations on XML documents are supported.
- XMLSpaces represents multiple dataspace servers, at different locations, as one logic dataspace.
- Distributed events are supported.

XMLSpaces is a well-described space-based computing implementation and has been extended also to the .NET platform [TLN04] which offers a bigger matching flexibility and is completely XML based. The project website can be found under `http://user.cs.tu-berlin.de/~tolk/xmlspaces` but only papers on XML spaces are available and not a source code.

A.103 XVSM

XVSM[197] [KRJ05], developed by the Space Based Computing Group of the Institute of Computer Languages at the Vienna University of Technology, combines the idea of tuple spaces with additional, flexible functionalities, called aspects, in order to provide exactly the functionality the user needs. Two different implementations are available:

- MozartSpaces: This is the open source Java implementation, where documentation and the source can be found on the project website under `http://www.mozartspaces.org`.
- XcoSpaces: is the .NET version which at the moment is available for students of the Space Based Computing Group of the Institute of Computer Languages at the Vienna University of Technology.

Important differentiation criteria to other implementations are [Sch08a]:

- XVSM offers the possibility (defining a dedicated server is also possible) to build the server out of the participating clients which share the entire data in the space amongst them.
- The communication protocol is independent in order to allow different applications to collaborate with each other without problems
- Additional features can be added into the space at runtime without being forced to restart the system.
- Elements can be ordered in the space through the use of coordinated data structures, e.g. queues or stacks.

XVSM is a very stable implementation which is continuously improved, e.g. [KRML08b] and information on it can be found on the project website[198] which offers also publications et cetera.

[197] eXtensible Virtual Shared Memory

[198] `http://www.complang.tuwien.ac.at/eva/research/researchpublications.html`

APPENDIX B

List of Figures

APPENDIX C

Acronyms

ACC	Agent Coordination Context
API	Application Programming Interface
ATIS	Agent Traveler Information Server
CAN	Content-Addressable Network
CCR	Microsoft Concurrency Coordination Runtime
Corso	Coordinated Shared Objects
Crudlet	creation, retrieval, updating, deletion, scheduling of life cycle activity, existence checking and generic templating of various objects in the context of business rules
DHT	Distibuted Hash Table
D-MANET	Disconnected Mobile Ad Hoc Networks
FIFO	First-In, First-Out
GONG	General Database Notification Gateway
Grinda	Grid and Linda
Gruple	Groovy and tuple

HTTPS	Hypertext Transfer Protocol Secure
ITS	Interface Tuple Space
ITS	Intelligent Transportation Services
JDBC	Java Database Connectivity
JEDI	Java Event-based Distributed Infrastructure
JION	JavaSpaces Implementation for Opportunistic Networks
JVM	Java Virtual Machine
JXTA	Juxtapose
KLAIM	Kernel Language for Agents Interaction and Mobility
KLAVA	Klaim in Java
LACIOS	Language for Agent Contextual Interaction in Open Systems
LBTS	Linearizable Byzantine Tuple Space
LGI	Law-Governed Infrastructure
LGL	Law-Governed Linda
LIFO	Last-In, First-Out
LIME	Linda in a Mobile Environmen
LINQ	Language Integrated Query
MANET	Mobile Ad Hoc Network
Melinda	Linda with multiple tuple spaces
MPI	Message Passing Interface
NAT	Network Address Translation
OTP	Open Telecom Platform

P2P	Peer-to-Peer
PDC	Parallel and Distributed Computing
P-Linda	Persistent Linda
PME	Programmable Matching Engine
PoliS	Poli Spaces
QoS	Quality of Service
RDF	Resource Description Framework
ReSpecT	Reaction Specification Tuples
RFID	Radio Frequency Identification
RHIO	Regional Health Information Organizations
RMI	Remote Method Invocation
RSA	Rivest, Shamir and Adleman
RSU	Road Side Units
SBC	Space Based Computing
SecOS	Secure Object Spaces
SOAP	Simple Object Access Protocol
SOP	Service-Orientated Programming
SQL	Structured Query Language
TCP	Transmission Control Protocol
TripCom	Triple Space Communication
TSTP	Triple Space Transfer Protocol

TuCSon	Tuple Centres Spread over the Network
UML	Unified Modelling Language
URI	Universal Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VLOS	Virtually Linda Operating System
VSM	Virtual Shared Memory
WCF	Windows Communication Foundation
WFMS	Worklflow Management System
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Networks
WPAN	Wireless Personal Area Network
WWAN	Wireless Wide Area Network
WWW	World Wide Web
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XQL	XML Query Language
XSL	Extensible Stylesheet Language
XVSM	eXtensible Virtual Shared Memory

Bibliography

[AAHC09] H. Artail, F. Al-Halabi, and A. Chehab. The design and implementation of an ad hoc network of mobile devices using the lime ii tuple-space framework. *Wireless Communications, IEEE*, 16(3):52 –59, 2009.

[AD99] Omicini; A. and L. Deis. Tuple centres for the coordination of internet agents. In *In Proc. of the 1999 ACM Symp. on Applied Computing (SAC 00*, pages 183–190. ACM Press, 1999.

[ADG99] M. Ancona, G. Dodero, and V. Gianuzzi. Ramses: A mobile computing system for field archaeology. volume 1707 of *Lecture Notes in Computer Science*, pages 222–233. Springer, 1999.

[ADNL15] Marina Andrić, Rocco De Nicola, and AlbertoLluch Lafuente. Replica-based high-performance tuple space computing. In Tom Holvoet and Mirko Viroli, editors, *Coordination Models and Languages*, volume 9037 of *Lecture Notes in Computer Science*, pages 3–18. Springer International Publishing, 2015.

[Ang03] W. Angleitner. Design und Implementierung eines graphischen Tools zur Visualisierung verteilter Datenstrukturen in CORSO-Spaces. Master's thesis, Fachhochschule Hagenberg, 2003.

[AR01] E. Astesiano and G. Reggio. Uml-spaces: a uml profile for distributed systems coordinated via tuple spaces. In *Autonomous Decentralized Systems*, pages 127–134, 2001.

[AS91] B. Anderson and D. Shasha. Persistent linda: Linda + transactions + query processing, 1991.

[Ass99] Scientific Computing Associates. Scientific Computing Associates: Virtual shared memory and the Paradise sys- tem for distributed computing. Technical report, 1999.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, December 2004.

[BACF08] A.N. Bessani, E.P. Alchieri, M. Correia, and S. Fraga. DepSpace : A Byzantine Fault-Tolerant Coordination Service. *Security*, pages 163–176, 2008.

[Bar10] M.S. Barisits. Design and Implementation of the next Generation XVSM Framework Operations , Coordination and Transactions. Master's thesis, Technical University of Vienna, 2010.

[BBDN+03] L. Bettini, V. Bono, R. De Nicola, G. Ferrari, D. Gorla, M. Loreti, Moggi. E., R. Pugliese, E. Tuosto, and B. Venneri. The klaim project: Theory and practice. In *GLOBAL COMPUTING: PROGRAMMING ENVIRONMENTS, LANGUAGES, SECURITY AND ANALYSIS OF SYSTEMS, VOLUME 2874 OF LNCS*, pages 88–150. Springer-Verlag, 2003.

[BC03] B. Brown and M. Chalmers. Tourism and mobile technology. In *Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*, ECSCW'03, pages 335–354. Kluwer Academic Publishers, 2003.

[BCKP95] R. Bagrodia, W. W. Chu, L. Kleinrock, and G. Popek. Vision, issues, and architecture for nomadic computing. *IEEE Personal Communications*, 2:14–27, 1995.

[BCP+] A. Belapurkar, A. Chakrabarti, H. Ponnapalli, N. Varadarajan, S. Padmanabhuni, and publisher = Sundarrajan, S. *Distributed systems security : issues, processes, and solutions*.

[BCP05] D. Balzarotti, P. Costa, and G.P. Picco. The lights tuple space frawework and its customization for context-aware applications, 2005.

[BDL+14] V. Boutin, C. Desdouits, M. Louvel, F. Pacull, M.I. Vergara-Gallego, O. Yaakoubi, C. Chomel, Q. Crignon, C. Duhoux, D. Genon-Catalot, L. Lefevre, Thanh Hung Pham, and Van Thang Pham. Energy optimisation using analytics and coordination, the example of lifts. In *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, pages 1–8, Sept 2014.

[BDN02] L. Bettini and R. De Nicola. A middleware for secure distributed tuple spaces, 2002.

[BDNP01] L. Bettini, R. De Nicola, and R. Pugliese. Klava: a java framework for distributed and mobile applications. In *Software Practice and Experience*, pages 32–1365, 2001.

[BdNPF98] L. Bettini, R. de Nicola, R. Pugliese, and G.L. Ferrari. Interactive mobile agents in x-klaim. In *Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 110–115, 1998.

[Ber04] T. Bernoulli. BEACON-LESS ROUTING IN MOBILE AD HOC NETWORKS. Master's thesis, Philosophisch-naturwissenschaftlichen Fakultät der Universität Bern, 2004.

[Bes06] A.N. Bessani. Bts: A byzantine fault-tolerant tuple space. In *Proceedings of the 21st ACM Symposium on Applied Computing - SAC 2006*, pages 429–433, 2006.

[Bet05] L. Bettini. Data Privacy in Tuple Space Based Mobile Agent Systems. *Electronic Notes in Theoretical Computer Science*, 128(5):3–16, 2005.

[BFK+11] S. Bessler, A. Fischer, E. Kühn, R. Mordinyi, and S. Tomic. Using tuplespaces to manage the storage and dissemination of spatial–temporal content. *Journal of Computer and System Sciences*, 77(2):322–331, March 2011.

[BGLZ02] N. Busi, R. Gorrieri, R. Lucchi, and G. Zavattaro. Secspaces: a data-driven coordination model for environments open to untrusted agents. In *Electronic Notes in Theoretical Computer Science*, 2002.

[Bie12] G. Bieber. Openwings. last visited: 02.08.2012.

[BLG12] A. Benchi, P. Launay, and F. Guidec. JION: A JavaSpaces Implementation for Opportunistic Networks. In *The Fourth International Conference on Future Computational Technologies and Applications (FUTURE COMPUTING 2012)*, pages 49–54, 2012.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, (5):34–43, 2001.

[BLL93] R.M. Butler, A.L. Leveton, and E.L. Lusk. p4-linda: a portable implementation of linda. In *High Performance Distributed Computing*, pages 50–58, 1993.

[BMSV] J. Brehm, C. Mueller-Schloer, and S. Voigt. An infospace paradigm for ad-hoc peer-to-peer communication.

[BO05] R. Beintner and H. Osius. Untersuchung des globus toolkit als grid computing middleware, 2005.

[BR02] C. Bryce and M. Razafimahefa, Ch.and Pawlak. Lana: An approach to programming autonomous systems. In *Proceedings of the 16th European Conference on Object-Oriented Programming*, pages 281–308. Springer-Verlag, 2002.

[BS95] D.E. Bakken and R.D. Schlichting. Supporting fault-tolerant parallel programming in Linda. *IEEE Transactions on Parallel and Distributed Systems*, 6(3):287–302, 1995.

[BS99] R. Buyya and LM SILVA. Parallel programming models and paradigms. *High Performance Cluster Computing: programming and applications. Melbourne: Pretince Hall*, 2:4–27, 1999.

[BST89] Henri E. Bal, Jennifer G. Steiner, and Andrew S. Tanenbaum. Programming languages for distributed computing systems. *ACM Computing Surveys*, pages 32–2, 1989.

[Bus05] Ch. Bussler. A minimal triple space computing architecture. In *In Proceedings of the 2nd WSMO Implementation Workshop*, 2005.

[Car89] N. Carriero. Linda in context. *Communications of the ACM*, 32(4):pp. 444–458, 1989.

[CB14] Justin Collins and Rajive Bagrodia. Mobile application development with melon. In Song Guo, Jaime Lloret, Pietro Manzoni, and Stefan Ruehrup, editors, *Ad-hoc, Mobile, and Wireless Networks*, volume 8487 of *Lecture Notes in Computer Science*, pages 265–278. Springer International Publishing, 2014.

[CCD+11] M. Ceriotti, M. Corra, L. D'Orazio, R. Doriguzzi, D. Facchin, S.T. Guna, G.P. Jesi, R.L. Cigno, L. Mottola, A.L. Murphy, M. Pescalli, G.P. Picco, D. Pregnolato, and C. Torghele. Is there light at the ends of the tunnel? wireless sensor networks for adaptive lighting in road tunnels. In *Information Processing in Sensor Networks (IPSN)*, pages 187–198, 2011.

[CDKB11] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair. *Distributed Systems: Concepts and Design (5th Edition)*. Addison Wesley, 5 edition, May 2011.

[CDNF01] G. Cugola, E. Di Nitto, and A. Fuggetta. The jedi event-based infrastructure and its application to the development of the opss wfms. *Software Engineering*, 27(9):827–850, 2001.

[CG86] N. Carriero and D. Gelernter. The S/Net's Linda kernel. *ACM Transactions on Computer Systems*, 4(2):110–129, 1986.

[CG90] N. Carriero and D. Gelernter. *How to write parallel programs: a first course*. MIT Press, Cambridge, MA, USA, 1990.

[CGG+05a] C. Curino, M. Giani, M. Giorgetta, A. Giusti, P. Milano, A.L. Murphy, and G.P. Picco. TinyLIME : Bridging Mobile and Sensor Networks through Middleware. *Communications*, (PerCom), 2005.

[CGG+05b] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. Murphy, and G. Picco. Mobile data collection in sensor networks: The TinyLime middleware. *Pervasive and Mobile Computing*, 1(4):446–469, 2005.

[CGG+05c] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A.L. Murphy, and G.P. Picco. Tinylime: bridging mobile and sensor networks through middleware. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pages 61–72, 2005.

[CGH97] N. Carriero, D. Gelernter, and S. Hupfer. Collaborative applications experience with the bauhaus coordination language. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, volume 1, pages 310–319, 1997.

[CGZ95] N. Carriero, D. Gelernter, and L. Zuck. Bauhaus Linda. In *ObjectBased Models and Languages for Concurrent Systems*, volume 924 of *LNCS*, pages 66–76. Springer-Verlag, 1995.

[Cia91] P. Ciancarini. POLIS: A Programming Model for Multiple Tuple Spaces. In *Proceedings of the 6th International Workshop on Software Specification and Design*, pages 44–51, 1991.

[Cia94] P. Ciancarini. Distributed programming with logic tuple spaces. *New Generation Computing*, 12:251–284, 1994.

[CLZ00] G. Cabri, L. Leonardi, and F. Zambonelli. Mars: a programmable coordination architecture for mobile agents, 2000.

[CM03] V.L. Chung and Ch. Mcdonald. Towards customisable tuple field matching in vlos, 2003.

[CM08a] S. Capizzi and A. Messina. Grinda: A tuple space service for the globus toolkit. In *Proceedings of the 8th international conference on Algorithms and Architectures for Parallel Processing*, pages 265–268. Springer-Verlag, 2008.

[CM08b] S. Capizzi and A. Messina. A tuple space service for large scale infrastructures. In *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 182–187, 2008.

[CMMP] P. Costa, L. Mottola, A.L. Murphy, and G.P. Picco. Tuple space middleware for wireless networks.

[CMMP06] P. Costa, L. Mottola, A.L. Murphy, and G.P. Picco. TeenyLIME: transiently shared tuple space middleware for wireless sensor networks. In *Proceedings of the international workshop on Middleware for sensor networks*, pages 43–48. ACM, 2006.

[CMMP07] P. Costa, L. Mottola, A.L. Murphy, and G.P. Picco. Programming Wireless Sensor Networks with the TeenyL IME Middleware. In *Proceedings of the 8th ACMIFIPUSENIX International Middleware Conference*, volume 4834 of *Lecture Notes in Computer Science*, pages 429–449. Springer, 2007.

[CMP+09] M. Ceriotti, L. Mottola, G.P. Picco, A.L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *Information Processing in Sensor Networks*, pages 277–288, 2009.

[CMT04] A. Charles, R. Menezes, and R. Tolksdorf. On the Implementation of SwarmLinda A Linda System Based on Swarm Intelligence (Extended Version). 2004.

[Col01] P. Colot. Klava : un package java pour programmer des applications distribuées avec mobilité de code, 2001.

[Col14] J. Collins. *Communication Paradigms for Mobile Ad Hoc Networks*. PhD thesis, UCLA, 2014.

[COZ00] M. Cremonini, A. Omicini, and F. Zambonelli. Coordination and Access Control in Open Distributed Agent Systems: The TuCSoN Approach. *Coordination Languages and Models*, 1906:99–114, 2000.

[CR97] P. Ciancarini and D. Rossi. Jada: coordination and communication for java agents. In *Mobile Object Systems: Towards the Programmable Internet*, pages 213–228. Springer-Verlag, 1997.

[CR02] P. Ciancarini and D. Rossi. Pagespace report f: Coordinating distributed applets with shade/java, 2002.

[CTZ02] P. Ciancarini, R. Tolksdorf, and F. Zambonelli. A Survey on Coordination Middleware for XML-centric Applications. *October*, 2002.

[CVF+07] D. Cerizza, E. Della Valle, D. Foxvog, R. Krummenacher, M. Murth, and Cefriel Politecnico Di Milano. Towards european patient summaries based on triple space computing, 2007.

[CVV04] B. Carbunar, M. T. Valente, and J. Vitek. Coordination and mobility in corelime. *Mathematical. Structures in Comp. Sci.*, 14:397–419, 2004.

[Daw12] M. Dawson. How enterprise will win back data control in 2013, 2012.

[DF96a] R. Drucker and A. Frank. A c++/linda model for distributed objects. In *Proceedings of the Seventh Israeli Conference on Computer Systems and Software Engineering*, pages 30–37, 1996.

[DF96b] R. Drucker and A. Frank. C++/Linda Model for Distributed Objects. *iccsse*, pages 30–37, 1996.

[DFBW98] N. Davies, A. Friday, G.S. Blair, and S. Wade. L2imbo: A Distributed Systems Platform for Mobile Computing. *Mobile Networks and Applications*, 3(2):143–156, 1998.

[DFWB98a] N. Davies, A. Friday, S.P. Wade, and G.S. Blair. An asynchronous distributed systems platform for heterogeneous environments. In *In Proceedings of the 8th ACM SIGOPS European*, pages 66–73. ACM Press, 1998.

[DFWB98b] N. Davies, A. Friday, S.P. Wade, and G.S. Blair. Limbo: A distributed systems platform for mobile computing. In *ACM Mobile Networks and Applications (MONET) - Special Issue on Protocols and Software Paradigms of Mobile Networks*, pages 143–156, 1998.

[DHN00] S. Ducasse, T. Hofmann, and O. Nierstrasz. OpenSpaces: An Object-Oriented Framework For Reconfigurable Coordination Spaces. In *Coordination Languages and Models*, volume 1906 of *LNCS*, pages 1–19, 2000.

[DNFP98] R. De Nicola, G.L. Ferrari, and R. Pugliese. Klaim: a kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.

[DNG10] S. De, S. Nandi, and D. Goswami. On performance improvement issues in unordered tuple space based mobile middleware. In *Proceedings of the India Conference (INDICON)*, pages 1–5, 2010.

[DNL00] R. De Nicola and M. Loreti. A modal logic for klaim, 2000.

[DO01] E. Denti and A. Omicini. Luce: A tuple-based coordination infrastructure for prolog and java agents. *Autonomous Agents and Multi-Agent Systems*, 4(1-2):139–141, 2001.

[Doe11] T. Doenz. Design and Implementation of the next Generation XVSM Framework Runtime, Protocol and API. Master's thesis, Technical University of Vienna, 2011.

[DOLA99] E. Denti, A. Omicini, L. Laboratorio, and I. Avanzata. Engineering multi-agent systems in luce. In *Proceedings of the ICLP 99 International Workshop on Multi-Agent Systems in Logic Programming (MAS 99)*, 1999.

[DOT00] E. Denti, A. Omicini, and V. Toschi. The luce coordination technology for mas design and development on the internet. In *Porto and Roman*, pages 305–310, 2000.

[DWFB97] N. Davies, S.P. Wade, A. Friday, and G.S. Blair. Limbo: A tuple space based platform for adaptive mobile applications. In *Proceedings of the International Conference on Open Distributed Processing/Distributed Platforms*, pages 291–302, 1997.

[DWR95] A. Douglas, A. Wood, and A. Rowstron. Linda implementation revisited. In *Proceedings of the 18th World Occam and Transputer User Group*, pages 125–138. IOS Press, 1995.

[EFGK03] P.Th. Eugster, P.A. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[EK07] M. Murth J. Riemer F. Schmied E. Kühn, G.Joskowicz. Spacebasedcomputing.org manifest. pages 1–12, 2007.

[ESZK04] J. Eberspächer, R. Schollmeier, St. Zöls, and G. Kunzmann. Structured p2p networks in mobile and fixed environments. In *IN PROC. OF THE INTERNATIONAL WORKING CONFERENCE ON PERFORMANCE MODELING AND EVALUATION OF HETEROGENEOUS NETWORKS*, 2004.

[EZCdrei92] St. Ericsson-ZENITH, ENSMP CRI, and Centre de recherche en informatique. *Les philosophies sur l'interaction des processus*. ENSMP, Fontainebleau, 1992.

[FAH99] E. Freeman, K. Arnold, and S. Hupfer. *JavaSpaces Principles, Patterns, and Practice*. Addison-Wesley Longman Ltd., Essex, UK, UK, 1st edition, 1999.

[Fen04] D. Fensel. Triple-space computing: Semantic web services based on persistent publication of information. In *In IFIP Internationall Conference on Intelligence in Communication Systems*, pages 43–53. Springer-Verlag, 2004.

[FKS+07] D. Fensel, R. Krummenacher, O. Shafiq, E. Kühn, J. Riemer, Y. Ding, and B. Draxler. Tsc – triple space computing. *e & i Elektrotechnik und Informationstechnik*, 124:31–38, 2007.

[For08] HIMSS EHR Implementation Toolkit Task Force. Building Enterprise Information Sharing: A Collection of Case Studies. Technical report, 2008.

[Fos02] I. Foster. What is the grid? a three point checklist, 2002.

[FP96] J.B.Jr. Fenwick and L.L. Pollock. Issues and experiences in implementing a distributed tuplespace, 1996.

[FRH04] Ch.L. Fok, G.C. Roman, and G. Hackmann. A lightweight coordination middleware for mobile computing. In *Proceedings of the 6th international conference on coordination models and languages*, pages 135–151. Springer-Verlag, 2004.

[FW14] Joerg Fritsch and Coral Walker. A novel weighted centroid localization algorithm based onrssi for an outdoor environment. pages 286–298, 2014.

[Gal97] O. Galibert. Ylc, a c++ linda system on top of pvm. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 1332 of *Lecture Notes in Computer Science*, pages 99–106. Springer Berlin / Heidelberg, 1997.

[Gar02] A.F. Garcia. Fault-Tolerance in Distributed Tuplespaces. pages 1–20, 2002.

[Gel85] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[Gel89] D. Gelernter. Multiple tuple spaces in Linda. *Science*, 365:20–27, 1989.

[Gel11] W. Gelbmann. Design and Implementation of LinqSpace. Master's thesis, Technical University of Vienna, 2011.

[GMT08] D. Graff, R. Menezes, and R. Tolksdorf. On the performance of swarm-based tuple organization in linda systems. In *Proceedings on the IEEE Congress on Evolutionary Computation*, pages 2709–2716, 2008.

[GSW97] R. Van Der Goot, J. Schae, and G.V. Wilson. Safer Tuple Spaces, 1997.

[GT02] A. Gibaud and P. Thomin. Communications directed by bound types in Linda: presentation and formal model. *IEEE Transactions on Parallel and Distributed Systems*, 13(8):828–843, 2002.

[Han02] P.B. Hansen. The origin of concurrent programming. chapter Joyce: a programming language for distributed systems, pages 464–492. Springer-Verlag New York, Inc., 2002.

[Haz93] T.K. Hazra. Occam channels and kernel linda. *Potentials, IEEE*, 12(1):15 –17, 1993.

[HB01] Mohamed Hefeeda and Bharat Bhargava. On mobile code security. Technical report, 2001.

[HD05] Manfred Hauswirth and Schahram Dustdar. Peer-to-peer: Grundlagen und architektur. *Datenbank-Spektrum*, 13:5–13, 2005.

[HJP02] K. A. Hawick, H. A. James, and L. H. Pritchard. Tuple-space based middleware for distributed computing, 2002.

[HR03] R. Handorean and G.C. Roman. Secure sharing of tuple spaces in ad hoc settings. 2003.

[Hup90] S.C. Hupfer. Melinda: Linda with multiple tuple spaces. *Technical Report YALEU /DCS/RR-766 Yale University*, 1990.

[Jel90] R. Jellinghaus. Eiffel linda: an object-oriented linda dialect. *SIGPLAN Not.*, 25(12):70–84, 1990.

[JHM11] R. Jones, A. Hosking, and E. Moss. *The Garbage Collection Handbook: The Art of Automatic Memory Management*. Chapman & Hall/CRC, 1st edition, 2011.

[Ji04] L. Ji. Coordination and P2P Computing. Master's thesis, University of Saskatchewan, 2004.

[Joh94] D.B. Johnson. Routing in ad hoc networks of mobile hosts. In *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, pages 158 –163, dec 1994.

[Jon12] N. Jones. Top 10 mobile technologies for 2012 and 2013, 2012.

[JR06] C. Julien and G.C. Roman. Egospaces: facilitating rapid development of context-aware mobile applications. *IEEE Transactions on Software Engineering,*, 32(5):281–298, 2006.

[JV04] S. Jagannathan and J. Vitek. Optimistic concurrency semantics for transactions in coordination languages. In Rocco Nicola, Gian-Luigi Ferrari, and Greg Meredith, editors, *Coordination Models and Languages*, volume 2949 of *Lecture Notes in Computer Science*, pages 183–198. Springer Berlin Heidelberg, 2004.

[JWM00] P. Jogalekar, M. Woodside, and Senior Member. Evaluating the scalability of distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 11:589–603, 2000.

[JXJY06] Y. Jiang, G. Xue, Z. Jia, and J. You. Dtuples: A distributed hash table based tuple space service for distributed coordination. In *Proceedings of the Fifth International Conference on Grid and Cooperative Computing*, pages 101–106, 2006.

[JYM08] A. Jaiantilal, J. Yifei, and S. Mishra. An evaluation of java rmi/javaspaces and ruby drb/rinda. In *Proceedings of the Performance, Computing and Communications Conference*, pages 127–134, 2008.

[KA07] M. Kinga and C. Adrian. Glinda - grid-based distributed linda system. In *Proceedings of the International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 349–352, 2007.

[Kar09] M. Karolus. Design and Implementation of XcoSpaces, the .Net Reference Implementation of XVSM - Coordination, Transactions and Communication. Master's thesis, Vienna University of Technology/Space-based Computing Group, 2009.

[KBM05] E. Kühn, M. Beinhart, and M. Murth. Improving data quality of mobile internet applications with an extensible virtual shared memory approach. In *IADIS International Conference on WWW/Internet 2005*. IADIS, 2005.

[KC12] E. Kühn and St. Craß. A coordination-based access control model for space-based computing. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 1560–1562, 2012.

[KCH14] E. Kuhn, S. Crass, and T. Hambock. Approaching coordination in distributed embedded applications with the peer model dsl. In *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*, pages 64–68, Aug 2014.

[Kla14] L. Klausner. Semantic xvsm design and implementation. Master's thesis, Technical University of Vienna, 2014.

[KLF04] D. Khushraj, O. Lassila, and T. Finin. stuples: semantic tuple spaces. In *Proceedings of The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 268–277, 2004.

[KMG+09] E. Kühn, R. Mordinyi, H.D. Goiss, T. Moser, S. Bessler, and S. Tomic. Integration of shareable containers with distributed hash tables for storage of structured and dynamic data. In *CISIS*, pages 866–871. IEEE Computer Society, 2009.

[KRJ05] E. Kühn, J. Riemer, and G. Joskowicz. Xvsm (extensible virtual shared memory) architecture and application, 2005.

[KRML08a] E. Kühn, J. Riemer, R. Mordinyi, and L. Lechner. Integration of XVSM spaces with the web to meet the challenging interaction demands in pervasive scenarios. *Ubiquitous Computing And Communication Journal (UbiCC), special issue on Coordination in Pervasive Environments*, 3, 2008.

[KRML08b] E. Kühn, J. Riemer, R. Mordinyi, and L. Lechner. Integration of XVSM spaces with the web to meet the challenging interaction demands in pervasive scenarios. *Ubiquitous Computing And Communication Journal (UbiCC), special issue on Coordination in Pervasive Environments*, 2008.

[Kro00] E. Kropf. Introduction à linda, 2000.

[Kue98] E. Kuehhn. How to approach the virtual shared memory paradigm, 1998.

[Kue03] E. Kuehn. The zero-delay data warehouse: mobilizing heterogeneous database. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 1035–1040. VLDB Endowment, 2003.

[Kue11] E. Kuehn. Slide presentation of space based computing paradigm. pages 1–24, 2011.

[LCX+01] T.J. Lehman, A. Cozzi, Y. Xiong, J. Gottschalk, V. Vasudevan, S. Landis, P. Davis, B. Khavar, and P. Bowman. Hitting the distributed computing sweet spot with TSpaces. *Computer Networks*, 35(4):457–472, 2001.

[Lel90] W. Leler. Linda meets unix. *Computer*, 23(2):43–54, 1990.

[Li01] S. Li. *Early Adopter JXTA: Peer-to-Peer Computing with Java*. Wrox Press Ltd., 2001.

[Li04] Y. Li. Reactive tuple space for a mobile agent platform. 2004.

[LKG+99] J.S. Lee, T.H. Kim, Yoon; G.S., J.E. Hong, S.D. Cha, and D.H. Bae. Developing distributed software systems by incorporating meta-object protocol (dimop) with unified modeling language (uml). In *Proceedings of The Fourth International Symposium on Autonomous Decentralized Systems*, pages 65–72, 1999.

[LKHK05] J.M. Leimeister, H. Krcmar, A. Horsch, and K. Kuhn. Mobile IT-Systeme im Gesundheitswesen, mobile Systeme für Patienten. *HMD – Praxis der Wirtschaftsinformatik*, 244:74–85, August 2005.

[LMW99] T.J. Lehman, S.W. McLaughry, and P. Wyckoff. T spaces: the next wave. In *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, 1999.

[LP05] Z. Li and M. Parashar. Comet: A scalable coordination space for decentralized distributed environments. In *In Proceedings of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 104–112. IEEE Computer Society Press, 2005.

[LPKL04] M. J. Lee, J. H. Park, S. J. Kang, and J. B. Lee. Multi-agent based home network management system with extended real-time tuple space. In *Proceedings of the 17th international conference on Innovations in applied artificial intelligence*, IEA/AIE'2004, pages 188–198. Springer Springer Verlag Inc, 2004.

[LR03] M. Leal and N.L.R. Rodriguez. LuaTS - A Reactive Event-Driven Tuple Space. *J. UCS*, 9(8):730–744, 2003.

[LS02] J.E. Larsen and S.H. Spring. *GLOBE*, volume 4863, pages 1–15. 2002.

[LT09] D. Lkhamsuren and Y. Tanaka. Padspace: A software architecture for the ad hoc federation of distributed visual components and web resources. In *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 113–120, 2009.

[LT10] D. Lkhamsuren and Y. Tanaka. Padlogspace: A prolog-based new software architecture for the service federation of web resources. In *Proceedings of the 2010 IET International Conference onFrontier Computing Theory, Technologies and Applications*, pages 390–395, 2010.

[Lwe10] B. Lwenstein. *Benchmarking of Middleware Systems: Evaluating and Comparing the Performance and Scalability of XVSM (MozartSpaces), JavaSpaces (GigaSpaces XAP) and J2EE (JBoss AS)*. VDM Verlag, 2010.

[Mas99] C. Mascolo. Mobis: A specification language for mobile systems, 1999.

[McD92] Ch. McDonald. Teaching concurrency with joyce and linda. *SIGCSE Bull.*, 24(1):46–52, 1992.

[MCE01] C. Mascolo, L. Capra, and W. Emmerich. An xml-based middleware for peer-to-peer computing. In *Proceedings of the First International Conference on Peer-to-Peer Computing*, pages 69–, 2001.

[MCW02] Ch. Mcdonald, V.L. Chung, and C. Wa. The development of a distributed capability system for vlos. In *Proceedings of the 7th Asia-Pacific Conference on Computer Systems Architecture - Volume 6, Conferences in Research and Practice in Information Technology Series*, 2002.

[MCZE02] C. Mascolo, L. Capra, St. Zachariadis, and W. Emmerich. Xmiddle: A data-sharing middleware for mobile computing, 2002.

[MKL+03] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical report, 2003.

[ML95] N. Minsky and J. Leichter. Law-governed linda communication model. In *Object-based Models and Languages for Concurrent Systems*, pages 125–146. Springer, 1995.

[MMMB09] Thomas Moser, Richard Mordinyi, A Mikula, and Stefan Biffl. Making expert knowledge explicit to facilitate tool support for integrating complex information systems in the atm domain. In *Intl. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS 2009)*, pages 90–97, 2009. Vortrag: Intl. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS 2009), Fukuoka, Japan; 2009-03-16 – 2009-03-19.

[MMU01] N. Minsky, Y.M. Minsky, and V. Ungureanu. Safe tuplespace-based coordination in multi agent systems, 2001.

[Mor10] R. Mordinyi. *Managing Complex and Dynamic Software Systems with Space-Based Computing*. PhD thesis, Vienna University of Technology/Space-based Computing Group, June 2010.

[MP04] A.L. Murphy and G.P. Picco. Using coordination middleware for location-aware computing: A lime case study. In *in Proc. of the 6 th Int. Conf. on Coordination Models and Languages (COORD04), LNCS 2949*, pages 263–278. Springer, 2004.

[MP06] A.L. Murphy and G.P. Picco. Using lime to support replication for availability in mobile ad hoc networks. In *In Proceedings of the 8 th International Conference on Coordination Models and Languages (COORDINATION 2006). Number 4038 in Lecture Notes on Computer Science*. Springer, 2006.

[MPR01] A.L. Murphy, G.P. Picco, and G.C. Roman. Lime: a middleware for physical and logical mobility. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 524–533, 2001.

[MPR06] A.L. Murphy, G.P. Picco, and G.C. Roman. LIME: A coordination model and middleware supporting mobility of hosts and agents. *ACM Transactions on Software Engineering and Methodology*, 15(3):279–328, 2006.

[MQZ06] M. Mamei, R. Quaglieri, and F. Zambonelli. Making tuple spaces physical with rfid tags. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 434–439, 2006.

[MT03] R. Menezes and R. Tolksdorf. A New Approach to Scalable Linda-systems Based on Swarms. *SAC*, pages 375–379, 2003.

[MTW01] R. Menezes, R. Tolksdorf, and A.M. Wood. Coordination of internet agents. chapter Scalability in Linda-like coordination systems, pages 299–319. Springer-Verlag, London, UK, UK, 2001.

[MW97] R. Menezes and A. Wood. Garbage collection in open distributed tuple space systems. In *In Proc. 15 th Brazilian Computer Networks Symposium — SBRC'97*, pages 525–543, 1997.

[MW98] R. Menezes and A. Wood. Ligia: A java based linda-like run-time system with garbage collection of tuple spaces. Technical report, 1998.

[MZ04] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the tota middleware. In *Proceedings of the Second IEEE Annual Conference on PerCom 2004*, pages 263–273, 2004.

[NAP+07] Lyndon Nixon, Olena Antonechko, Elena Paslaru, Bontas Simperl, and Robert Tolksdorf. Towards semantic tuplespace computing: The semantic web spaces system. In *In 22nd ACM Symposium on Applied Computing*, 2007.

[NBCdSFCL07] A. Neves-Bessani, M. Correia, J. da Silva Fraga, and L. Cheuk Lung. An efficient byzantine-resilient tuple space, 2007.

[NCOA10] E. Nardini, M. Casadei, A. Omicini, and A.L. Ater. A Self-Organising Infrastructure for Chemical-Semantic Coordination : Experiments in TuCSoN. In Andrea Omicini and Mirko Viroli, editors, *WOA 2010 Eleventh National Workshop From Objects to Agents*, volume 621 of *CEUR Workshop Proceedings*, pages 117–125. Sun SITE Central Europe, RWTH Aachen University, 2010.

[NCV+07] L.J. B. Nixon, D. Cerizza, E. Della Valle, E. Simperl, R. Krummenacher, and Reto. Enabling collaborative ehealth through triplespace computing. In *Proceedings of the 16th IEEE International Workshops on Enabling Technolo-*

gies: Infrastructure for Collaborative Enterprises, WETICE '07, pages 80–85, Washington, DC, USA, 2007. IEEE Computer Society.

[Neu03] R. Neubauer. Skalierbarkeit und Ausfallsicherheit für CORBA. Master's thesis, Technical University of Vienna, 2003.

[NIS10] A. Nour, R. Isidro, and C. Solís. Ambient-prisma: Ambients in mobile aspect-oriented software architecture. *J. Syst. Softw.*, 83(6):937–958, 2010.

[NOVS11] E. Nardini, A. Omicini, M. Viroli, and M.I. Schumacher. Coordinating e-health systems with tucson semantic tuple centres. *SIGAPP Appl. Comput. Rev.*, 11(2):43–53, 2011.

[NPS03] E. Newcomb, T. Pashley, and J. Stasko. Mobile computing in the retail arena. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 337–344. ACM, 2003.

[NSKMR08] L.J.B. Nixon, E. Simperl, R. Krummenacher, and F. Martin-Recuerda. Tuplespace-based computing for the Semantic Web: a survey of the state-of-the-art. *The Knowledge Engineering Review*, 23(02):181–212, 2008.

[NVCO10] E. Nardini, M. Viroli, M. Casadei, and A. Omicini. A self-organising infrastructure for chemical-semantic coordination: Experiments in tucson, 2010.

[NVP10] E. Nardini, M. Viroli, and E. Panzavolta. Coordination in open and dynamic environments with TuCSoN semantic tuple centres. *Proceedings of the 2010 ACM Symposium on Applied Computing SAC 10*, page 2037, 2010.

[OG02] P. Obreiter and G. Graef. Towards scalability in tuple spaces. In *Proceedings of the 2002 Symposium on Applied Computing*, pages 344–350. ACM, 2002.

[OH05] M. Oriol and M. Hicks. Tagged sets: A secure and transparent coordination medium. In *Proceedings of the 7th Int. Conf. on Coordination Models and Languages*, pages 252–267. Springer-Verlag, 2005.

[OM12] A. Omicini and St. Mariani. Tucson: Tuple centres spread over the network basics, 2012.

[OM13] A. Omicini and St. Mariani. The tucson coordination model & technology - a guide, 2013.

[Omi06] A. Omicini. Formal respect in the a&a perspective. In *University of Málaga, Spain. Proceedings*. Post-proceedings, 2006.

[Ora01] A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[ORRV03] A. Omicini, A. Ricci, G. Rimassa, and M. Viroli. Integrating objective and subjective coordination in fipa: A roadmap to tucson, 2003.

[OZ98a] A. Omicini and F. Zambonelli. Tucson: a coordination model for mobile information agents. In *Proceedings of the 1st workshop on innovative internet innovation systems*, pages 177–187, 1998.

[OZ98b] A. Omicini and F. Zambonelli. Tucson: a coordination model for mobile information agents. In *IN PROCEEDINGS OF THE 1ST WORKSHOP ON INNOVATIVE INTERNET INFORMATION SYSTEMS*, pages 177–187, 1998.

[Pap05] White Paper. GigaSpaces Data Grid - Caching. *Design*, pages 1–14, 2005.

[PB05] G.P. Picco and D. Balzarotti. Lights: A lightweight, customizable tuple space supporting context-aware applications. In *Proceedings of the 20 the ACM Symposium on Applied Computing*. ACM Press, 2005.

[PCBB07] J. Pauty, P. Couderc, M. Banatre, and Y. Berbers. Geo-linda: a geometry aware distributed tuple space. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications*, pages 370–377, 2007.

[Per08] Ch. E. Perkins. *Ad Hoc Networking*. Addison-Wesley Professional, 1 edition, 2008.

[Pla06] C. Plattner. *Ganymed: A Platform for Database Replication*. ETH, 2006.

[PLBB15] Carlo Pinciroli, Adam Lee-Brown, and Giovanni Beltrame. Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms, 2015.

[PM91] J.N. Pinakis and C.S. McDonald. The inclusion of the linda tuple space operations in a pascal-based concurrent language. In *Proceedings of the 14th Australian Computer Science Conference*, pages 1–11. 1991.

[PM99] GP Picco and AL Murphy. LIME: Linda meets mobility. *Proceedings of the 21st*, 1999.

[PMR99] G.P. Picco, A.L. Murphy, and G.C. Roman. Lime: Linda meets mobility. In *Proceedings of the 1999 International Conference on Software Engineering*, pages 368–377, 1999.

[PMR00] G.P. Picco, A.L. Murphy, and G.C. Roman. Developing mobile computing applications with lime. In *Proceedings of the 2000 International Conference on Software Engineering*, pages 766 –769, 2000.

[Pri02] Ch. Wachenfeld St. Prinzleve, S. Tepper. Verteilte programmierung eines agentensystems mit lime - evaluationsbericht, 2002.

[Pro08] M. Prostler. Design and Implementation of MozartSpaces , the Java Reference Implemention of XVSM Timeout Handling , Notifications and Aspects. Master's thesis, Vienna University of Technology/Space-based Computing Group, 2008.

[PS95] D. Plainfosse and M. Shapiro. A survey of distributed garbage collection techniques, 1995.

[PTW01] S. Prinzleve, Ch. Tepper, and St. Wachenfeld. Adaptive cluster computing using javaspaces, 2001.

[PTW02] S. Prinzleve, Ch. Tepper, and St. Wachenfeld. Verteilte programmierung eines agentensystems mit lime evaluationsbericht, 2002.

[Ran10] P. Rantanen. Database replication an overview of replication techniques in common database systems, 2010.

[Rau14] D. Rauch. Peerspace.net implementing and evaluating the peer model with focus on api usability. Master's thesis, Technical University of Vienna, 2014.

[RCVS04] G. Russello, M. Chaudron, and M. Van Steen. Gspace: Tailorable data distribution in shared data space system. Technical report, 2004.

[RCVS05] G. Russello, M. Chaudron, and M. Van Steen. Dynamically adapting tuple replication for managing availability in a shared data space. In *In Coordination Model and Languages - COORDINATION 05), volume 3454 of LNCS*. Springer, 2005.

[RDD+08] G. Russello, Ch. Dong, N. Dulay, M. Chaudron, and M. Van Steen. Encrypted shared data spaces. In *Proceedings of the 10th international conference on Coordination models and languages*, COORDINATION 08, Berlin, Heidelberg, 2008. Springer-Verlag.

[Res12] Dimensional Research. The impact of mobile devices on information security: A survey of it professionals, 2012.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *IN PROC. ACM SIGCOMM 2001*, pages 161–172, 2001.

[RHCN02] M. Roman, C. Hess, R. Cerqueira, and K. Nahrstedt. Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, 1:74–83, 2002.

[RO02] A. Ricci and A. Omicini. Agent coordination contexts: Experiments in tucson, 2002.

[ROD01] A. Ricci, A. Omicini, and E. Denti. Enlightened agents in tucson, 2001.

[Row97] A. Rowstron. Using asynchronous tuple-space access primitives (BONITA primitives) for process co-ordination. *Coordination Languages and Models*, pages 426–429, 1997.

[RW97] A.I.T. Rowstron and A.M. Wood. Bonita: a set of tuple space primitives for distributed coordination. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, volume 1, pages 379–388, 1997.

[Sat96] M. Satyanarayanan. Fundamental challenges in mobile computing. In *In Fifteenth ACM Symposium on Principles of Distributed Computing*, pages 1–7, 1996.

[Sau08] J. Saunier. *Les communications multi-parties et leur régulation dans les systèmes multi-agents : modèle et support*. PhD thesis, Université Paris-Dauphine, 2008.

[Sch01] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 101–102, 2001.

[Sch07] A. Schwind. Space-based web services: Konzepte und prototypische implementierung mit linda-spaces. Master's thesis, University of Stuttgart, 2007.

[Sch08a] T. Scheller. Classification of Space Based Computing Systems, 2008.

[Sch08b] T. Scheller. Design and Implementation of XcoSpaces, the .Net Reference Implementation of XVSM - Core Architecture and Aspects. Master's thesis, Vienna University of Technology/Space-based Computing Group, 2008.

[Sch08c] Ch. Schreiber. Design and Implementation of MozartSpaces, the Java Reference Implementation of XVSM Custom Coordinators, Transactions and XML protocol. Master's thesis, Vienna University of Technology/Space-based Computing Group, 2008.

[Sci05] Scientific Computing Associates Inc. *Linda User Guide*, 2005.

[Sek09] M. Seki. druby and rinda: Implementation and application of distributed ruby and its parallel coordination mechanism. *International Journal of Parallel Programming*, 37(1):37–57, 2009.

[SF02] D. Schoder and K. Fischbach. Peer-to-peer anwendungsbereiche und herausforderungen, 2002.

[SHBHDR11] A. Sheng-Hai, L. Byung-Hyug, and S. Dong-Ryeol. A survey of intelligent transportation systems. In *Computational Intelligence, Communication Systems and Networks (CICSyN), 2011 Third International Conference on*, pages 332 –337, july 2011.

[Slu07] T.A. Sluga. Modern c++ implementation of the linda coordination language for distributed applications. pages 1 – 69, 2007.

[Son03] K. Song. Analyzing transactions on jini and javaspace. Master's thesis, Arizona State University, 2003.

[SP89] G. Sutcliffe and J. Pinakis. Prolog-linda : An embedding of linda in muprolog, 1989.

[Sut93] G. Sutcliffe. Prolog-d-linda v2 : A new embedding of linda in sicstus prolog. In *Proceedings of the Workshop on Blackboard-based Logic Programming*, pages 105–117, 1993.

[TG01a] R. Tolksdorf and D. Glaubitz. Coordinating web-based systems with documents in xmlspaces. In *Proceedings of the Sixth IFCIS International Conference on Cooperative Information Systems (CoopIS 2001), number LNCS 2172*, pages 356–370. Springer Verlag, 2001.

[TG01b] R. Tolksdorf and D. Glaubitz. Xmlspaces for coordination in web-based systems. In *Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE '01, pages 322–327, Washington, DC, USA, 2001. IEEE Computer Society.

[Tho02] P. Thompson. Ruple: an xml space implementation. 2002.

[TKA10] S. Tyagi, M. A. Khan, and A.Q Ansari. *RFID Data Management, Radio Frequency Identification Fundamentals and Applications Bringing Research to Practice*. 2010.

[TLN04] R. Tolksdorf, F. Liebsch, and D. Nguyen. Xmlspaces.net: An extensible tuplespace as xml middleware. In *In Report B 03-08, Free University Berlin, ftp://ftp.inf.fu-berlin.de/pub/reports/tr-b-0308.pdf, 2003. Open Research Questions in SOA 5-25 and Loose Coupling in Service Oriented Architectures*, 2004.

[TM03] R. Tolksdorf and R. Menezes. Using swarm intelligence in linda systems. In *Proceedings of the Fourth International Workshop Engineering Societies in the Agents World ESAW03*. Springer Verlag, 2003.

[TN04] R. Tolksdorf and L.J.B. Nixon. Semantic web spaces: A concrete use case and implementation requirements. 2004.

[TN08] K. Teymourian and L.J.B. Nixon. Efficient content location in massively distributed triplespaces. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2008 Workshops, OTM Confederated International Workshops and Posters, ADI, AWeSoMe, COMBEK, EI2N, IWSSA, MONET, OnToContent + QSI, ORM, PerSys, RDDS, SEMELS, and SWWS 2008, Monterrey, Mexico, November 9-14,*, volume 5333 of *Lecture Notes in Computer Science*, pages 947–956. Springer, 2008.

[TS90] H. Takeda and T. Satoh. An accelerating processor for relational operations. In *Proceedings of the International Conference on Databases, Parallel Architectures and Their Applications*, page 559, 1990.

[UDI09] N.I. Udzir, S. Demesie, and H. Ibrahim. Garbage collection in lindacap. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, pages 104–112. ACM, 2009.

[Unk12] Unknown. Blitz project. last visited: 02.08.2012.

[UOdlT+14] Visara Urovi, Alex C. Olivieri, Albert Brugués de la Torre, Stefano Bromuri, Nicoletta Fornara, and Michael Schumacher. Secure p2p cross-community health record exchange in ihe compatible systems. *International Journal on Artificial Intelligence Tools*, 23(01):1440006, 2014.

[UWJ07] N.I. Udzir, A.M. Wood, and J.L. Jacob. Coordination with multicapabilities. *Sci. Comput. Program.*, 64(2):205–222, 2007.

[VBO03] J. Vitek, B. Bryce, and M. Oriol. Coordinating processes with secure spaces, 2003.

[Wat14] H. Watzke. Lifecycle and memory management for extensible virtual shared memory (xvsm). Master's thesis, Technical University of Vienna, 2014.

[WCC01] G. Wells, P. Clayton, and A. Chalmers. Extending linda to simplify application development. pages 108–114, 2001.

[WCC04] G. C. Wells, A. G. Chalmers, and P. G. Clayton. Linda implementations in java for concurrent systems: Research articles. *Concurr. Comput. : Pract. Exper.*, 16(10):1005–1022, August 2004.

[Wei91] M. Weiser. The Computer for the 21st Century. *Scientific American*, 1991.

[Wei93] M. Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7):75–84, July 1993.

[Wei11] M. Weitzel. Einsatzmöglichkeiten für Technologien des Semantic Web im Rahmen der strategischen Planung intermodaler Seehafenhinterland-Transport-Ketten. Master's thesis, Universität Rostock Fakultät für Informatik und Elektrotechnik Institut für Informatik, 2011.

[Wel05] G. Wells. Coordination languages: Back to the future with linda. In *Proceedings of WCAT05*, pages 87–98, 2005.

[WG06] V. Weerackody and L. Gonzalez. Performance of satellite communications on the move systems in the presence of antenna pointing errors. In *Proceedings of the 2006 IEEE conference on Military communications*, MILCOM'06, pages 3145–3151, Piscataway, NJ, USA, 2006. IEEE Press.

[WGH07] S. Weinbrenner, A. Giemza, and H.U. Hoppe. Engineering heterogeneous distributed learning environments using tuple spaces as an architectural platform. In *Proceedings on the Seventh IEEE International Conference on Advanced Learning Technologies*, pages 434–436, 2007.

[Wil12] A. Wilkinson. Pylinda. last visited: 02.08.2012.

[Win11] M. Winkler. XIDS An XVSM-Based Collaborative Intrusion Detection System. Master's thesis, Technical University of Vienna, 2011.

[WMLF98] P. Wyckoff, S.W. McLaughry, T.J. Lehman, and D.A. Ford. T spaces. *IBM Systems Journal*, 37(3):454 –474, 1998.

[WWF+10] U. Wickramasinghe, C. Wickramarachchi, P. Fernando, D. Sumanasena, S. Perera, and S. Weerawarana. Bissa: Empowering web gadget communication with tuple spaces. In *Gateway Computing Environments Workshop (GCE), 2010*, pages 1–8, 2010.

[YF96] C.K. Yuen and M.D. Feng. Use of active objects in balinda k for system programming. In *Proceedings of the Second International Conference on the Algorithms and Architectures for Parallel Processing*, pages 438–445, 1996.

[YFY93] C.K. Yuen, M.D. Feng, and J.J. Yee. Speculative parallelism in balinda lisp. In *Proceedings of the Fifth International Conference on Computing and Information*, pages 261–265, 1993.

[YMA+10] F. Yang, H. Masuhara, T. Aotani, F. Nielson, and H.R. Nielson. Aspectke*: Security aspects with program analysis for distributed systems, 2010.

[YW90] C.K. Yuen and W.F. Wong. Balinda lisp: a parallel list-processing language. In *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, pages 618–624, 1990.

[ZBH09] M. Zargayouna, F. Balbo, and S. Haddad. Agents secure interaction in data driven languages. In *Proceedings of the Second Multi-Agent Logics, Languages, and Organisations Federated Workshops*, pages 1–8. Springer-Verlag, 2009.

[ZBH10] M. Zargayouna, F. Balbo, and S. Haddad. Data driven language for agents secure interaction. In *Proceedings of the Second international conference on Languages, Methodologies, and Development Tools for Multi-Agent Systems*, pages 72–91. Springer-Verlag, 2010.

[ZBS09] M. Zargayouna, F. Balbo, and G. Scémama. A data-oriented coordination language for distributed transportation applications. In *Proceedings of the Third KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pages 283–292. Springer-Verlag, 2009.

[ZEN92] S.E. ZENITH. A rationale for programming with ease. In *Research Directions in High-Level Parallel Programming Languages*, pages 147–156. Springer-Verlag, 1992.