D2.1 – Dealing with data intensiveness and cognitive complexity in contemporary collaboration and decision making settings: A state of the art survey

Due date of deliverable: 28 February 2011
Actual submission date: 14 March 2011
Responsible Partner: CTI
Contributing Partners: CTI, FHG, NEO, UPM, UOL

Nature: □ Report    □ Prototype    □ Demonstrator    □ Other

Dissemination Level:
□ PU : Public
□ PP : Restricted to other programme participants (including the Commission Services)
□ RE : Restricted to a group specified by the consortium (including the Commission Services)
□ CO : Confidential, only for members of the consortium (including the Commission Services)

Keyword List: Scalable high performance data mining, data mining towards sense-making, collaboration support, decision making, integration, data intensiveness, cognitive complexity, state-of-the-art

The Dicode project (dicode-project.eu) is funded by the European Commission, Information Society and Media Directorate General, under the FP7 Cooperation programme (ICT/SO 4.3: Intelligent Information Management).
The Dicode Consortium

Research Academic Computer Technology Institute (CTI), Greece (coordinator)

University of Leeds (UOL), UK

Fraunhofer-Gesellschaft Zur Foerderung Der Angewandten Forschung E.V. (FHG), Germany

Universidad Politecnica de Madrid (UPM), Spain

neofonie GmbH (NEO), Germany

Image Analysis Limited (IMA), UK

Biomedical Research Foundation, Academy of Athens (BRF), Greece

Publicis Frankfurt GmbH (PUB), Germany
Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Status</th>
<th>Modifications made by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21-12-2010</td>
<td>First Draft version</td>
<td>Isabel Drost, Spyros Christodoulou, Manolis Tzagarakis, Guillermo de la Calle Velasco, Axel Poigné, Joerg Kindermann, Styliani Kleanthous</td>
</tr>
<tr>
<td>2</td>
<td>14-01-2011</td>
<td>Second Draft version</td>
<td>Manolis Tzagarakis, Isabel Drost, Axel Poigné, Nikos Karacapilidis</td>
</tr>
<tr>
<td>3</td>
<td>12-02-2011</td>
<td>Third Draft version</td>
<td>Isabel Drost, Spyros Christodoulou, Manolis Tzagarakis, Guillermo de la Calle Velasco, Axel Poigné, Joerg Kindermann, Vania Dimitrova, Styliani Kleanthous, Stefan Rueping, Nikos Karacapilidis</td>
</tr>
<tr>
<td>4</td>
<td>28-02-2011</td>
<td>Version sent to internal reviewers</td>
<td>Manolis Tzagarakis</td>
</tr>
<tr>
<td>5</td>
<td>09-03-2011</td>
<td>Reviewers’ comments incorporated (version sent to SC)</td>
<td>Manolis Tzagarakis</td>
</tr>
<tr>
<td>6</td>
<td>14-03-2011</td>
<td>Final (approved by SC, sent to the Project Officer)</td>
<td>Manolis Tzagarakis</td>
</tr>
</tbody>
</table>

Deliverable manager
- Manolis Tzagarakis, CTI

List of Contributors
- Isabel Drost, NEO
- Axel Poigné, FHG
- Dennis Wegener, FHG
- Joerg Kindermann, FHG
- Stefan Rueping, FHG
- Nikos Karacapilidis, CTI
- Manolis Tzagarakis, CTI
- Spyros Christodoulou, CTI
- Styliani Kleanthous, UOL
- Vania Dimitrova, UOL
- Guillermo de la Calle Velasco, UPM

List of Evaluators
- Lydia Lau, UOL
- Duncan Russell, IMA
Summary

This deliverable reviews the state-of-the-art on the three key research directions of the Dicode project, namely scalable high-performance data mining, data mining towards sense-making of real-world multi-faceted data, and collaboration and decision making support. A review of the state-of-the-art on integration technologies, which underpin the Dicode architecture, is also included. The methodologies, tools and approaches discussed in the deliverable are considered with respect to the information overload and cognitive complexity dimensions, which are inherent in Dicode. The deliverable aims to devise a roadmap and draw useful conclusions concerning the exploitation and advancement of existing technologies for the needs of project.
# Table of Contents

1. **Introduction** .................................................................................................................. 7  
   1.1. Information overload and cognitive complexity in contemporary business, scientific and social settings .................................................. 7  
   1.2. Aim and Structure of this Deliverable ..................................................................... 8  

2. **Decision making support: on the need of synergy between human and machine reasoning** .................................................................................................................. 10  
   2.1. On the evolution of decision making support technologies .................................. 10  
   2.2. Prominent decision making support technologies ................................................. 12  
   2.3. Paving the way to the Dicode approach ................................................................. 13  

3. **Scalable High-performance Data Mining** .................................................................. 15  
   3.1. Introduction ............................................................................................................. 15  
   3.2. Distributed Data Mining ...................................................................................... 15  
   3.3. Local Thread-Level Distribution ........................................................................ 17  
   3.3.1. Thread-level Distribution for CPUs .................................................................. 17  
   3.3.2. Thread-level Distribution for GPUs ............................................................... 18  
   3.3.3. Mixed CPU - GPU Models ............................................................................. 19  
   3.3.4. Data Mining Algorithm for Thread-level Distribution .................................. 19  
   3.3.5. Summary ......................................................................................................... 23  
   3.4. Cluster Level Distribution ................................................................................... 23  
   3.4.1. MapReduce ..................................................................................................... 23  
   3.4.2. DryadLINQ ..................................................................................................... 26  
   3.4.3. Alternative Approaches .................................................................................. 28  
   3.5. Higher level applications ..................................................................................... 29  
   3.5.1. Machine Learning Suites ............................................................................... 29  
   3.5.2. Data preparation ............................................................................................. 31  
   3.6. Implications for Dicode ....................................................................................... 31  

4. **Data Mining towards Sense-making of Real-world Multi-faceted Data** ................. 33  
   4.1. Introduction .......................................................................................................... 33  
   4.2. The Data Mining Process ................................................................................... 33  
   4.2.1. Characteristics of Data Mining ....................................................................... 34  
   4.3. Machine learning and data mining technologies ................................................. 35  
   4.4. Applications in Text Mining ................................................................................. 37  
   4.4.1. Text indexing and relevance feedback ......................................................... 37  
   4.4.2. Text annotation ............................................................................................... 38  
   4.4.3. Text classification ........................................................................................... 38  
   4.4.4. Text clustering ............................................................................................... 39  
   4.5. Subgroup Discovery ............................................................................................. 39  
   4.6. Data Mining and Business Process Management .............................................. 40  
   4.7. Data Mining and Data Analysis Processes for Research .................................... 42  
   4.8. Implications for Dicode ....................................................................................... 43  

5. **Collaboration and Decision Making Support** .......................................................... 45  
   5.1. Introduction .......................................................................................................... 45  
   5.2. Mind mapping ....................................................................................................... 46
5.3. File sharing and collaborative editing................................................................. 48
5.4. Social networking.................................................................................................. 51
5.5. Note taking and annotation................................................................................... 52
5.6. Project and task management................................................................................ 53
5.7. Argumentative collaboration.................................................................................. 55
5.8. Collaboration tools overview and implications for Dicode ..................................... 57
5.9. Community Monitoring and Adaptive Collaboration Support............................. 60
  5.9.1. Approaches for Community Modelling............................................................ 61
  5.9.2. Discovering connections.................................................................................. 61
  5.9.3. Modelling interests.......................................................................................... 62
  5.9.4. Modelling expertise......................................................................................... 62
  5.9.5. Community graph models.............................................................................. 62
  5.9.6. Community patterns....................................................................................... 63
  5.9.7. Approaches for community support............................................................... 63
  5.9.8. Implications for Dicode .................................................................................. 64

6. Integration issues ........................................................................................................ 65
  6.1. Introduction........................................................................................................... 65
  6.2. Data Integration ..................................................................................................... 65
    6.2.1. Centralized approaches .................................................................................. 66
    6.2.2. Distributed / Federated approaches ............................................................... 66
  6.3. Applications Integration ....................................................................................... 68
    6.3.1. Serviced-Oriented Architecture..................................................................... 70
    6.3.2. Examples of technologies to implement SOA ............................................... 75
    6.3.3. Examples of SOA technologies in the field of Bioinformatics ....................... 76
    6.3.4. Resource Oriented Architecture: REST...................................................... 77
    6.3.5. Examples of REST applications .................................................................... 79
    6.3.6. Comparison: SOA vs. REST ....................................................................... 80
  6.4. Implications for Dicode ......................................................................................... 81

7. Summary and Conclusions ...................................................................................... 83

8. References............................................................................................................... 86
1. Introduction

1.1. Information overload and cognitive complexity in contemporary business, scientific and social settings

Information overload has become a major problem for today’s organisations. While incoming data is rapidly increasing, making sense and filtering what is important for the current situation, becomes difficult and time consuming. This becomes an even bigger problem where collaboration and decision making is taking place. These settings are often associated with huge, ever-increasing amount of multiple types of data, obtained from diverse and distributed sources. In many cases, the raw information is so overwhelming that stakeholders are often at a loss to know even where to begin to make sense of it. “Big Data”, as this problem is called (Economist, 2010), can negatively affect the effectiveness of decision making in an organisation (Schick, 1990) and create stress and cognitive overload to its stakeholders (Kirsh, 2000).

In addition, this data may vary in terms of subjectivity and importance, ranging from individual opinions and estimations to broadly accepted practices and indisputable measurements and scientific results. Different formats exist some of which require domain experts to interpret and some require specific software. Besides, it is nowadays easier to get the data in than out. Large volumes of data can be effortlessly added to a database (e.g. in transaction processing). The problems start when we want to consider and exploit the accumulated data in the open, which may have been collected over a few weeks or months, and meaningfully analyze them towards making a decision. Thus, in complex settings such as a community of clinical researchers and bio-scientists or in a contemporary business such as a marketing and consultancy company, being able to pull only the information relevant to a current problem solving scenario and efficiently share, interpret and use these information for decision making, becomes a challenge when the right tools and information systems are missing (Batra, 2007). Admittedly, when things get complex, we need to identify, understand and exploit data patterns; we need to aggregate large volumes of data from multiple sources, and then mine it for insights that would never emerge from manual inspection or analysis of any single data source. In other words, the pathologies of “big data” are primarily those of analysis.

In the settings under consideration, the way that data will be structured for query and analysis, as well as the way that tools will be designed to handle them efficiently are of great importance and certainly set a big research challenge. Generally speaking, information management related tasks are directly linked with a community’s or organisation’s context. To make information work more productive, these tasks need to be streamlined and automated (in cases of predictable or repeatable steps). As reveals from an IDC’s survey covering 600 U.S. companies of diverse size and domain\(^1\):

- Searching for and analyzing information both consume 24% of the typical information worker’s time (9.5 and 9.6 hours per week, respectively);
- Information workers spend 8.3 hours per week in gathering information for documents and 6.8 hours per week to file and organize them;
- Information workers waste 3.5 hours in searching for information that is never found (caused by poor search, or lack of integrated access to all the enterprise’s collections

\(^1\) IDC (www.idc.com), The Hidden Costs of Information Work, White Paper, March 2005
of information, or lack of good content), and 2.3 hours in acquiring archived records
with little or no automation.

Such findings clearly indicate that information management costs too much when it is not
well organized and meaningfully automated. Furthermore, a central problem for complex
organisations is one of coping with uncertainty, but uncertainty can be reduced through
information (Thompson, 1967). Thus, the problem now becomes not how to bring this
information in the organisation but how to retrieve the information needed for a task, from
the information the organisation already possesses (Schick, 1990). They also call for
investments in innovative software that reduces or eliminates time wasted, reduces
management overheads, streamlines collaborative processes, and automates the overall
workflow. Return on such investments can be both tangible (e.g. time or money saved) and
intangible (e.g. more valuable information, easier extraction of hidden information, increase
of information workers' satisfaction and creativity, improved collaboration).

Overall, such innovative solutions have to face two major imperatives:

- They need to exploit the information growth by ensuring a flexible, adaptable and
  scalable information and computation infrastructure; new approaches – taking
  advantage of distributed computing and clusters of computational resources - are
  needed for structured and unstructured data search, data pre-processing, database
  analytics, resource pooling, and storage optimization (Economist, 2010).

- They need to exploit the competences of all stakeholders and information workers to
  meaningfully confront various information management issues (Beardsley et al.,
  2006), (e.g. information characterization, classification, presentation, retention,
  storage, disposal etc.); dealing with data-intensive and cognitively complex settings is
  not a technical problem alone.

Moreover, when dealing with complex scientific and contemporary evidence-based
organisations, information and data need to take a central role with technologies, systems
and services built around them. Stakeholders should be provided means to effectively
retrieve, share and exploit what is relevant for them at the point of need from the
information available in the organisation (Economist, 2010).

1.2. Aim and Structure of this Deliverable

In this deliverable, we review the state-of-the-art on the three key research directions of the
Dicode project, namely scalable high-performance data mining, data mining towards sense-
making of real-world multi-faceted data, and collaboration and decision making support. A
review of the state-of-the-art on integration technologies, which underpin the Dicode
architecture, is also included. The aim of this review is to devise a roadmap on how existing
technologies must be adapted and used in order to successfully address the issues that arise
in the context of Dicode. In particular, the review evaluates existing technologies with
respect to their ability to tackle information overload and cognitive complexity issues, which
are endemic to the project’s use cases. For each technology, strengths and shortcomings are
highlighted, and recommendations on its use in the project are formulated.

The deliverable is structured as follows: Section 2 outlines the overall decision making
context, pointing to the need of synergy between human and machine reasoning. Sections 3,
4 and 5 present in detail the state-of-the-art on the three key research directions of the Dicode
project (scalable high-performance data mining, data mining towards sense-making of real-
world multi-faceted data, and collaboration and decision making support, respectively);
strengths and weaknesses of the related technologies are highlighted, while
recommendations related to their potential in addressing the information overload and cognitive complexity issues of Dicode are also formulated. Section 6 is devoted to the state-of-the-art on integration technologies. Finally, Section 7 concludes the deliverable by summarizing key findings and identifying major target outcomes, as far as the project’s research and applied work is concerned.
2. Decision making support: on the need of synergy between human and machine reasoning

Decision making is ubiquitous in the contemporary organizational processes. According to Simon (1977), it comprises three principal phases: identifying problematic situations or opportunities that call for decisions (intelligence phase), inventing or developing possible courses of action and testing of their feasibility (design phase), and selecting a certain course of action to be followed (choice phase). As noted in (McLaughlin, 1995), “successful organizations out-decide their competitors in at least three ways: they make better decisions, they make decisions faster, and they implement more decisions”. The quality, speed and realization of the decision making can be increased when the right information is available to the right persons, at the right time and in the right form.

Software systems aiming at offering support in the decision making process first appeared in the late 1960s. Decision Support Systems (DSS) have been defined as “interactive computer-based systems, which help decision makers utilize data and models to solve unstructured problems” (Gorry and Scott Morton, 1971). According to Keen and Scott Morton (1978), DSS “couple the intellectual resources of individuals with the capabilities of the computer to improve the quality of decisions”. As pointed out in (Pearson and Shim, 1995), research on this field during the past three decades has mainly focused on how information technology can improve the efficiency with which a user makes a decision, and can improve the effectiveness of that decision. More specifically, quoting (Shim et al., 2002), much research and practical effort has been conducted in developing technologies to be exploited in DSS components for “sophisticated database management capabilities with access to internal and external data, information and knowledge, powerful modelling functions accessed by a model management system, and powerful, yet simple user interface designs that enable interactive queries, reporting and graphing functions”.

2.1. On the evolution of decision making support technologies

Generally speaking, technologies for decision making support are assembled from four basic components, namely data, models, knowledge, and user interface (Turban and Aronson, 2001; Mora et al., 2003). It is the particular combination of the above components that defines the features and the functionality of a DSS (these components are related to the subsystems a DSS consists of). More specifically, the first of them is associated with the data management subsystem, which handles the extracting, organizing and archiving an organization’s internal and external data. Development of the appropriate database, database management system (DBMS), data directory, and query facility are the major issues to be addressed in this subsystem. The second component relates to the model management subsystem, which provides the DSS’s analytical capabilities. Development of a quantitative model base and tools to manage the creation, updating, integration and execution of its constituent models are the critical issues here. In turn, the knowledge component is associated with the knowledge-based management subsystem that characterizes more advanced DSS, and provides the knowledge and expertise needed to solve aspects of an ill-structured problem. Such an aid may concern issues related to the decision process per se (steps to be followed), the models to be built, and the manipulation of the uncertainty inherent in the problem. Finally, the last component is associated with the user interface subsystem, which covers all aspects of communication and interaction between a user and the DSS.
From a conceptual perspective, a DSS may be (Power, 2002): (i) **communication-driven** (network and communication technologies are used to facilitate collaboration, more than one person work on the same task resulting in a faster and more effective decision making process), (ii) **data-driven** (accessing, analyzing and manipulation of a time-series of internal (company) and external data are supported), (iii) **document-driven** (refers to searching, retrieving and manipulating a large number of unstructured data), (iv) **knowledge-driven** (providing specialized problem solving knowledge and decision making in particular domains), or (v) **model-driven** (analytical and optimization tools are used for decision problem modelling; statistical, financial, optimization or simulation models are used).

In the 1970s and early 1980s, decision making support technologies were customarily focused on model development and problem analysis, while over the last two decades the related research has evolved to include additional concepts and views (Forgionne et al., 2002; Shim et al., 2002). These include Group Decision Support Systems (GDSS), attempting to provide evaluation of ideas in a brainstorming setting where the decision responsibility is shared by a number of users (DeSanctis and Galleupe, 1987; Arnott and Pervan, 2005), Executive Information Systems (EIS), extending the scope of a DSS from personal or small group use to a corporate level (to provide reporting about the nature of an organization to management; emphasis is on graphical displays and easy-to-use user interfaces), and Knowledge-Based DSS, aiming at offering enhanced support to decision makers by encapsulating techniques from the disciplines of Artificial Intelligence (AI).

The advent of Internet and Web, as well as modern communication technology has resulted to the broadening of the organizational environment. Courtney (2001) suggested that DSS researchers have to embrace a much more comprehensive view of the organizational decision making context and accordingly develop systems that are able to handle “softer” information. What started to evolve in the last few years is that issues related to the mental models of decision makers, expressing their organizational, personal and technical perspectives of the problem under consideration, are critical and have to be carefully addressed.

Summarizing, it is clear that the introduction of DSS received great attention from the beginning, since these systems were heading to important developments such as the integration of interactive systems for managers and professionals, the achievement of user-friendly environments, and the provision of a suitable framework for the handling of semi-structured and unstructured tasks. However, research on this area, having over-dealt with technological and definition issues (e.g., the differences between a DSS and an Expert System or an Executive Information System), has de-emphasized other major issues in improving decision making (Alter, 1992). These issues include work structuring in order to improve coordination, use of communication technology to make decision making more efficient and effective, enforcing of rules and procedures for achieving consistency, and (semi)automation of data processing in data intensive decision making situations. Angehrn and Jelassi (1994) have urged the DSS community to further consider the conceptual, methodological and application-oriented aspects of the problem. Conceptual focus is associated with the consideration of the nature of individual and organizational decision making processes, methodological focus with the integration of existing computer-based tools, techniques and systems into the human decision making context, and application-oriented focus with the consideration of the real organizational needs. Considering the above aspects, a series of prominent technologies has been proposed and evolved.
2.2. Prominent decision making support technologies

Data warehouses, on-line analytical processing, data mining and web-based DSS have been broadly recognized as technologies playing a prominent role in the development of current and future DSS (Shim et al., 2002; Turban and Aronson, 2001). More specifically, data warehouses are subject-oriented, integrated, time-variant, non-volatile collections of data (O’Brien and Marakas, 2009); they provide the infrastructure that enables businesses to extract, cleanse, and store vast amounts of corporate data from operational systems for efficient and accurate responses to user queries (Kimball and Ross, 2002) and empower knowledge workers with information that allows them to make decisions based on a solid foundation of facts (Devlin, 1997). However, only a part of the required knowledge can be represented in computers; a data warehouse does not provide adequate support for knowledge intensive queries in the organization. As argued in (Nemati et al., 2002), what is needed is “a new generation of knowledge-enabled systems that provides the infrastructure required to capture, enhance, store, organize, leverage, analyze, and disseminate not only data and information but also knowledge”.

Data stored in a data warehouse are usually analyzed with the aid of on-line analytical processing (OLAP) tools. OLAP has been defined as a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user (Thomsen, 2002). According to the database technology for building a data warehouse, two basic types of OLAP tools are distinguished, namely Multidimensional OLAP (MOLAP) and Relational OLAP (ROLAP). Each of these types has its own advantages and disadvantages, while a third one, namely Hybrid OLAP (HOLAP), attempts to combine the advantages of the first two.

The power of the above applications in processing vast amounts of data can be significantly augmented by data mining applications. Such tools are built on concepts and techniques from AI and Statistics (such as Case-Based Reasoning, Data Visualization, Fuzzy Analysis, and Neural Networks), aiming at providing a more sophisticated data analysis by discovering patterns of data and inferring data content relationships and rules from them (Fayyad et al., 1996; Han and Chang, 2002). Contemporary data mining applications can learn from the previous history of the investigated system, as well as shape and test hypotheses about the rules which this system acts upon. In case that appropriate knowledge has been formulated, they can be incorporated into a DSS to aid managers make better decisions. The ever-growing body of information that exists in the Web is being also exploited through the use of the above technologies to support a series of decision making settings.

At the same time, the Web environment became a widely adopted development and delivery platform. Web-Based DSS deliver information and/or tools to a decision maker through a Web browser that is accessing the Internet or a corporate intranet. The computer server that

More specifically, MOLAP is the more traditional type of OLAP analysis, where data is stored in a multidimensional cube. It is more appropriate for cubes with frequent use and when there exists a necessity for rapid query response. It can quickly perform complex calculations, since all of them have been pre-generated (at the creation of the cube). However, it is often limited in the amount of data it can handle (this is related to the amount of data that can be included in the cube), and requires additional investment from an organization. On the other hand, ROLAP performs dynamic analysis of data stored in a relational database and leverages its functionalities. It does not use pre-calculated data cubes; instead, it intercepts the query and poses the question to the standard relational database and its tables in order to bring back the data required to answer the question. Its disadvantages are slow response and limited scalability (depending on the technology architecture that is utilized). However, compared to MOLAP, it supports larger user groups and greater amounts of data and is often used when these capacities are crucial.
is hosting the DSS application is linked to the user's computer by a network with the TCP/IP protocol. Web-Based DSS can be communications, data, document, knowledge, or model driven, or built following a hybrid approach. Depending on the network type they are based on, they can provide specific decision making capabilities to diverse user types, such as to managers over an intranet, customers and suppliers over an extranet, or to any stakeholder over the Internet (Shim et al., 2002).

Two additional DSS technologies, falling into the Artificial Intelligence (AI) discipline, are Rule-Based Systems (Hayes-Roth, 1985; Ignizio, 1990) and Case-Based Reasoning (Kolodner, 1993; Watson, 1997). Rule-Based Systems (RBS) do not represent knowledge in a declarative and static way; instead, they do so through a set of “if-then” rules that indicate what has to be done or concluded at a specific instance of the problem under consideration (i.e., given a set of facts). Reasoning is performed through either forward chaining, where using the initial facts the system exploits rules to draw new conclusions or take certain actions (data-driven approach), or backward chaining, where the system attempts to satisfy the goals in the goal stack by finding rules that can conclude the information needed and trying to satisfy the “if” parts of those rules (goal-driven approach). Generally speaking, RBS are of practical importance for problems for which the related knowledge can be expressed in the form of the above rules and the problem area is not large. If there are too many rules, RBS become difficult to maintain and are characterized by low performance.

On the other hand, according to the Case-Based Reasoning (CBR) technology, expertise is encoded in a library of past cases (not in rules). Typically, each case comprises a description of a certain instance of the problem and its solution or outcome (the knowledge and the reasoning process followed to reach the solution is not explicitly recorded). To solve a new problem instance, a matching of it against past cases is performed (according to diverse similarity measures), the aim being to retrieve similar cases and exploit their solutions. These solutions may be revised for the new instance of the problem, while the new instance and its final solution shape a new case to be stored in the case base. CBR received growing interest in the last decade, both from an academic and commercial point of view. Its suitability to a decision making context depends on a set of parameters related to whether records of previously solved problem instances exist, historical cases are viewed as a valuable asset which has to be retained, and exploitation of previous experiences is considered as useful and is of common practice.

2.3. Paving the way to the Dicode approach

The above technologies certainly facilitate diverse aspects of decision making. Although there exist certain limitations in their suitability (Karacapilidis, 2006), they may aid DSS users to make better and faster decisions. However, there is still room for further developing the conceptual, methodological and application-oriented aspects of the problem. One critical point that is still missing is a holistic perspective on the issue of decision making. This originates out of the growing need to develop applications by following a more human-centric (not problem-centric) view, in order to appropriately address the requirements of the contemporary, knowledge-intensive organization’s employees. Such requirements stem from the fact that decision making has also to be considered as a social process that principally involves human interaction (Smoliar, 2003). The structuring and management of this interaction requires the appropriate technological support and has to be explicitly embedded in the system. The above requirements, together with the ones imposed by the way decision makers work and collaborate today, delineate a set of challenges for further decision support technology development.
More specifically, as argued in (Prahalad and Hamel, 1990), “a firm’s only advantage in today’s business environment is its ability to leverage and utilize its knowledge”. While a firm comprises individuals and a set of definable objectified resources, its most strategically important feature is its body of collective knowledge (Spender, 1996). Such knowledge resides in an evolving set of assets including the employees, structure, culture and processes of the organization. Of these, employee knowledge, and particularly tacit knowledge is identified as the dominant one, which is decisive at all mental levels and has to be fully exploited (Nonaka, 1994). Such exploitation refers to the transformation of tacit knowledge to codified information, which is considered as a core process for economic activity and development (Cohendet and Steinmueller, 2000). The above advocate the adoption of a knowledge-based decision-making view (Holsapple and Whinston, 1996), which should delineate the future development of decision support technologies. According to this view, decisions should be considered as pieces of descriptive or procedural knowledge referring to an action commitment. In such a way, the decision making process is able to produce new knowledge, such as evidence justifying or challenging an alternative or practices to be followed or avoided after the evaluation of a decision, thus providing a refined understanding of the problem. On the other hand, in a decision making context the knowledge base of facts and routines alters, since it has to reflect the ever-changing external environment and internal structures of the organization (Bhatt and Zaveri, 2002). Knowledge management activities such as knowledge elicitation, representation and distribution influence the creation of the decision models to be adopted, thus enhancing the decision making process (Bolloju et al., 2002).

The abovementioned integrated consideration of decision making and knowledge management can be further strengthened by the incorporation of features enabling decision makers to perform argumentation and experimentations on the issues raised. Many collaborative decision making problems have to be solved through dialoguing and argumentation among a group of people (van Eemeren et al., 1996; Provis, 2004). In such contexts, conflicts of interest are unavoidable and support for achieving consensus and compromise is required. Each decision maker may formulate and put forward his/her own position that fulfills some goals with a specific acceptance level. Moreover, he/she may have arguments in favor or against alternative solutions, as well as preferences and constraints imposed on them. Depending on the role and the goals of each decision maker, subjective estimates of the problem should be taken into consideration. Independently of the model used for decision making, argumentation is valuable in shaping a common understanding of the problem. It can provide the means to decide which parts of the information brought up by the decision makers will finally be the input to the model used. Moreover, argumentation may stimulate the participation of decision makers and encourage constructive criticism. To address the above category of requirements, a user-friendly discourse-based decision support environment should be developed.

The above discussion paves the way to the intelligent information management foreseen in the context of the Dicode project, which exploits and builds on the synergy of human (collective) and machine (artificial) intelligence, by giving them equal importance. Approaches concerning the latter, and in particular scalable high-performance data mining and data mining towards sense-making of real-world multi-faceted data, are discussed in detail in Sections 3 and 4, respectively. Approaches primarily concerning the human intelligence, aiming to facilitate and enhance collaboration and decision making support in diverse settings, are considered in Section 5.
3. Scalable High-performance Data Mining

3.1. Introduction

The goal of Dicode is to bring Data Mining to the realm of huge datasets. A growing amount of data can easily be stored and processed in recent years with the advent of data analysis software stacks that are deployed on low-cost commodity hardware. Solutions, proprietary or free, are available to facilitate data analysis in large scale applications. Usually distributing computation to separate compute nodes plays a major role in scaling data analysis algorithms.

There are two main strategies for scaling data analysis algorithms:

- To reduce the size of data by sampling in a preprocessing step to obtain a much smaller subset of the original data to which further data analysis is applied.
- To distribute analysis tasks for concurrent evaluation either by exploiting local parallelism, as provided by multicores or multiprocessors, or by distributing data and tasks to a computer network.

Both strategies have advantages and disadvantages.

The obvious advantage of sampling is that it reduces costs for the subsequent data analysis though the cost of obtaining a sample is proportional to the size of the data. However in contrast to other data reduction techniques, sampling does not require a complete pass through the data. It is possible to determine a sufficient sample size for estimating a given function within a specified degree of error (central limit theorem). On the other hand, sampling increases the variance of the estimates that are constructed from the data. While this is usually not a problem for “simple” statistics such as the mean or correlation, it quickly becomes problematic when more complex models are to be inferred from the data. In the extreme case where one is interested in rare events, the probability of having a rare event represented in a small-size sample is very small in the first place, such that no meaningful model can be extracted. A different problem is the situation where there is high dependency between different examples because there is a global structure in the data. This is for example prevalent in real-world data sets from the web domain, which often can be viewed as instances of one huge web graph. While some approaches for sampling from graphs exist, a significant reduction of the size of the graph can usually only be achieved by breaking the global structure (Leskovec and Faloutsos, 2006).

In contrast, distributed analysis often scales well even to the extent of petabytes. Disadvantages here are that there is often no simple procedure to proceed from a sequential algorithm to a distributed one and that not every data mining task is suited for distribution.

Dicode will take advantage of both strategies for high-performance data mining but in this report focus is on distributed data mining since sampling techniques are sufficiently covered by the current literature.

3.2. Distributed Data Mining

Three levels of distribution can be distinguished:

Local thread-level distribution: The speed of any single processor has not been increasing substantially in recent years. However the number of individual cores placed in one machine is rapidly increasing with Oracle offering CPUs with as much as 512 threads per machine
This development clearly indicates that even when using a single machine there is a clear need for parallel code execution. Especially compute-intensive algorithms that are comparably independent of data access benefit from these kinds of systems. However in the realm of data analysis the main bottleneck usually is not the CPU found in a system but the available IO capacity.

Another variant of locally distributing work is to use GPUs for computation. They offer a wide range of facilities for distributed matrix computations that can easily be of use in the area of machine learning and data mining. However GPU based distribution also suffers from the problem of limited IO capacity rendering them useless when faced with vast amount of data. For compute intensive tasks these chips still provide a great alternative for speeding up computation.

**Cluster level distribution:** A common setup when dealing with growing amount of data is to distribute computations on multiple separate machines, communicating and synchronising their work through networking facilities. There are various libraries available in the area of distributed cluster computations that make it easy to express distribution patterns in code. Especially well known in the distributed computing community are tools such as PVM or MPI. Both libraries offer a high degree of freedom when designing a software system. On the other hand they require developers to have a deep understanding of distributed programming models and handle most intricacies of data distribution and communication patterns themselves.

A decade ago Google came up with the idea of using a highly constrained pattern for formulating distributed programs on top of a distributed file system called Google File System (GFS): Reusing the Map()/Reduce() functions well known from functional programming it was observed that expressing analysis steps as map/reduce functions leads to the desirable property of having to express code executed during the map phase data-independent from all other map functions. Same applies to the reduce phase. This property makes it possible to automatically distribute code running in map and reduce steps thus leading to a decreased amount of work necessary to come up with distributed programs. All networking, data distribution, disk IO can be handled in a uniform way. As a result it becomes simple to roll out improvements to an entire map/reduce cluster without affecting its users.

**Grid level distribution:** In the context of scientific computations, a vast amount of work has been done towards building grid-level distributed systems. These are mostly used for collaborating on a scale that is out of scope for Dicode.

In the area of data-parallel, cluster-level distribution, traditional databases provide an all-in-one solution to the problem of data analysis. They are very appealing as in general developers are familiar with using SQL databases. In addition they usually provide a complete solution for efficient data storage, analysis and querying. With SQL there is a standard query interface that can be used across systems making it easy to port applications developed from one database to another. In addition queries to the database usually are optimised internally for performance and speed.

However the seemingly perfect solution comes with one grave caveat. Scaling databases is usually trivial when upgrading the machines running the database to bigger boxes. However scaling out at a very large scale is not a trivial problem, usually involving very high costs either for maintenance or for software licenses in case of professional systems.

As a result providers of search engines who had to deal with growing amounts of data as early as 2003 started looking into different systems for data analysis. In general they are made up of three building blocks: a software artefact for distributing data storage; a low-
level component for expressing data analysis jobs; and finally a higher-level component that provides easier and faster access to common analysis steps (Figure 1).

![Data-Parallel Computation](image)

**Figure 1:** Building blocks of data-parallel computation

### 3.3. Local Thread-Level Distribution

The advent of multicore processors and multiprocessor machines demands for parallel code execution in order to exploit the potential on offer. Furthermore, the availability of GPUs as general purpose computing devices as well as the increasingly tighter integration of CPUs and GPUs marks another development.

#### 3.3.1. Thread-level Distribution for CPUs

Multicores (and multiprocessors) provide a boost of performance to be noticed by improved response times in particular for CPU-intensive tasks in that many of these may execute in parallel. The downside is that operating systems and applications need adjustment in that execution of multiple threads must be organised and optimised without putting too much burden on the programmer. Programming multi-threaded code is inherently more error-prone but the gain in throughput may be considerable. Designing for multicores implies identifying tasks within a program that can run in parallel without interfering with each other and organising execution so that a maximal number of these is executed in parallel if feasible.

A number of parallel programming models supports programming for multiple cores/multiprocessors, notably OpenMP (http://openmp.org/wp/). OpenMP is a shared memory programming model. It offers a number of primitives for organising thread execution. The primitive are typically added as pragmas to sequential code as in

```c
#pragma omp parallel for
    for (i = 0; i < N; i++)
    a[i] = 2 * i;
```

Here the value of a large array is initialised in parallel, each thread doing a portion of the work. A welcome effect is that sequential code can often be instrumented incrementally, with the sequential code running unaffected if the pragmas are skipped. OpenMP is platform-independent but it requires compiler support. Compilers are provided for C/C++ and Fortran.
Grand Central Dispatch (http://de.wikipedia.org/wiki/Grand_CentralDispatch) is an alternative developed by Apple and is provided as library libdispatch released under the Apache License. The general idea is that tasks that can run in parallel are queued and then scheduled for execution depending on the available resources. Tasks are specified as blocks, an approximation of closures, i.e. anonymous functions that have access to local variables as in the following parallelised version of a for loop:

```c
dispatch_apply(count, dispatch_get_global_queue(0, 0),
    {^(size_t i) results[i] = do_work(data, i); } );
```

The block

```
{^(size_t i) results[i] = do_work(data, i); }
```

is dispatched to a global queue to run count times. Grand central dispatch provides the optimal number of threads given the current hardware and system load. Several queues can be handled and synchronisation can be imposed on tasks.

There are a number of other approaches to extend standard thread mechanisms such as POSIX (http://en.wikipedia.org/wiki/Multi-core_processor). Common idea is to ease multi-threaded programming by avoiding the most obvious pitfalls.

### 3.3.2. Thread-level Distribution for GPUs

GPU Computing or GPGPU for General Purpose Computing on GPU has found its way into fields where computational power based on SIMD (Single Instruction, Multiple Data) offers great benefit in such diverse fields as oil exploration, scientific image processing, linear algebra, statistics, image processing, and data mining. Nvidia's CUDA (http://www.nvidia.com/object/cuda_home_new.html) platform is the most widely adopted programming model for GPU computing, with OpenCL (http://www.khronos.org/opencl/) also being offered as an open standard. Architectural components of CUDA are:

- **Kernel**: basic unit of program that can run on a GPU at a time. A kernel is capable of executing N number of threads in parallel.

- **Threads**: CUDA threads are very lightweight with little overhead so that they can be switched almost instantaneously. They are organized in the form of grid-block-thread hierarchy. A grid can contain N number of blocks. Each block is capable of having M number of threads. Threads within a block can cooperate with each other using shared memory in the same block. All threads within a block are executed concurrently.

- **Memory**: Three kinds of memories are supported with decreasing access speed:
  - **Per Thread Local Memory**: Each thread has its private local memory.
  - **Per Block Shared Memory**: threads within a block have access to.
  - **Global Memory**: all threads have access.

- **Streaming**: Speed of data transfer between host and GPU is by order of magnitude slower the internal bus speeds of the GPU. On the other hand, internal memory is restricted. Streaming allows transferring data in parallel with kernel execution resulting in a high throughput if data transfer and kernel execution are properly aligned.
3.3.3. Mixed CPU - GPU Models

The design space for local thread-level distribution is delimited by several parameters:

- The inherent parallelism of the algorithm
- The memory available for GPU
- The communication bandwidth between GPU and CPU
- The number of multicores/multiprocessors
- The available fast memory on the CPU

Within a high degree of data parallelism, GPUs offer little advantage since otherwise more time may be spent on the exchange of data between CPU host and GPU than is needed for executing the same computations on the CPU only. Rare memory on the GPU is the next bottleneck, though streaming concepts can help if there is a balance between computation time on the GPU and data transfer time on the CPU-GPU busses, which depends on the bandwidth. Other typical phenomenon is that the breadth of parallel execution decreases with computation time. Then a break-even point has to be found when to migrate from GPU to CPU if transfer time starts to dominate computation time on the GPU. Fast memory on the CPU with a sufficient number of threads managed by multicores/multiprocessors may as well diminish the benefits of using GPUs even if the degree of potential parallelism in an algorithm is high.

The upshot is that exploitation of the design space depends on the concrete (data mining) algorithm and on the particular kind of data. GPUs may offer computational power exceeding using CPUs only by order of magnitudes and then additionally scale extremely well for multi-GPU clusters, but may fail on the same scale for algorithms with too many causal dependencies involved. For mixed models, the combination of CUDA and openMP seems the choice of most algorithm designers.

3.3.4. Data Mining Algorithm for Thread-level Distribution

A growing number of data mining algorithms have been adapted for multicores and GPUs. These fall into two broad categories

- Algorithms with inherent data parallelisation such as, e.g., k-means
- Algorithms based on matrix manipulation.

A (non-exhaustive) list of references is provided below summarising the main results.

**Targeting Multicores:**

<table>
<thead>
<tr>
<th>Reference</th>
<th>Summary</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Chu et al., 2006)</td>
<td>Statistical Query model is written in a certain “summation form,” which allows them to be easily parallelised on multicore computers. Based on Map-reduce.</td>
<td>Locally Weighted Linear Regression, Gaussian Discriminative Analysis, k-means, Logistic Regression, Neural Network, Principal Components Analysis, Independent Component, Expectation Maximization, Support Vector Machine</td>
</tr>
<tr>
<td>(Rao et al., 2010)</td>
<td>Based on memory-mapped IO using OpenMP</td>
<td>k-means</td>
</tr>
</tbody>
</table>
Targeting GPUs:

<table>
<thead>
<tr>
<th>Reference</th>
<th>Summary</th>
<th>Algorithm</th>
<th>Data sets</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Yang et al., 2010)</td>
<td>Novel approach to data representation for computing the sparse matrix-vector multiplication kernel on GPUs</td>
<td>PageRank, HITS and Random Walk with Restart</td>
<td>User link relationship graphs from Flickr, LiveJournal and Youtube and a webpage link relationship graph from Wikipedia</td>
<td>PageRank: 13x and 31x speedup, HITS: 11x to 28x speedup, Random Walk with Restart: 13x to 37x speedup</td>
</tr>
<tr>
<td>(Archuleta et al., 2009)</td>
<td>Characterization of a MapReduce-based data-mining application on a general-purpose GPU. Provides eight general performance characterizations based on a data-mining application in neuroscience in order to guide the GPGPU programmer towards optimal performance within the design space.</td>
<td>Frequent Episode Mining</td>
<td>Database of 393019 letters. Episodes up to length 3 are considered.</td>
<td>The impact of episode length, chosen algorithm, and GPU card are compared.</td>
</tr>
<tr>
<td>(Ma et al., 2009a)</td>
<td>CUDA code generated from sequential reduction loop(s) with some information about the parameters using program analysis and additional optimizations based on LLVM.</td>
<td>k-means clustering</td>
<td>Not known</td>
<td>Speedup of 50 over the CPU version, 20 when the data movement time is included</td>
</tr>
<tr>
<td>(Sharp, 2008)</td>
<td>Decision forest is mapped to a 2D texture array and trees are evaluated with non-branching pixel shaders. Training of decision trees involves new GPU hardware features to compute histograms with a combination of pixel and vertex shaders</td>
<td>Object class recognition via randomized decision forests</td>
<td>Images</td>
<td>Speedup of 100</td>
</tr>
<tr>
<td>(He et al., 2008)</td>
<td>Design and implementation a MapReduce framework for GPUs</td>
<td>Applications for web data analysis: String Match, Inverted Index, Similarity Score, Matrix Multiplication, Page View Count, and Page View Rank</td>
<td>Not indicated, for every application data sets qualified as small, medium, and large are used</td>
<td>Up to 7x faster than its CPU counterpart with optimised multicore algorithms</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>(Ma et al., 2009b)</td>
<td>Automatic generation of parallel code based on instrumented sequential loops</td>
<td>k-means clustering, EM clustering, and Principal Component Analysis (PCA)</td>
<td>Not indicated</td>
<td>Speedup between 20 and 50</td>
</tr>
<tr>
<td>(Fang et al., 2009)</td>
<td>Frequent itemset mining based on bitmap representation</td>
<td>Apriori</td>
<td>T40I10D100K, Chess, and Retail (from FIMI’03 repository)</td>
<td>Speedup of up to two orders of magnitude over optimized CPU Apriori implementations. However, FP-growth is 4 – 16 times faster than GPU implementations.</td>
</tr>
<tr>
<td>(Yan et al., 2009)</td>
<td>Novel data partitioning scheme are proposed to effectively reduce the memory cost. Data streaming is used to handle extremely large datasets.</td>
<td>Dirichlet Allocation (LDA) models, Collapsed Gibbs Sampling (CGS) and Collapsed Variational Bayesian (CVB)</td>
<td>Text datasets retrieved from the UCI Machine Learning Repository</td>
<td>26x speedup for CGS and 196x speedup for CVB on a GPU with 30 multiprocessor s</td>
</tr>
</tbody>
</table>
# Targeting Mixed Models:

<table>
<thead>
<tr>
<th>Reference</th>
<th>Summary</th>
<th>Algorithms</th>
<th>Technology</th>
<th>Data sets</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(He et al., 2008)</td>
<td>Uses a Map-Reduce generic framework for a combination of GPU and Multicores.</td>
<td>String Match, Inverted Index, Similarity Score, Matrix Multiplication, Page View Count, Page View Rank</td>
<td>CUDA for GPU. Technology for multicores is not discussed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| (Yan et al., 2009) | Novel data partitioning scheme are proposed to effectively reduce the memory cost. Data use data streaming is used to handle extremely large datasets. | Dirichlet Allocation (LDA) models,Collapsed Gibbs Sampling (CGS) and Collapsed Variational Bayesian (CVB) | CUDA, OpenMP, OpenMPI (see Section 3.4.3) | Text datasets retrieved from the UCI Machine Learning Repository | 26x speedup for CGS and 196x speedup for CVB on a GPU with 30 multiprocessors |

A number of CRAN packages are available porting R algorithms to GPUs (http://cran.r-project.org/web/views/HighPerformanceComputing.html):

- The gputools package (cran.r-project.org/web/packages/gputools/) by Buckner provides several common data mining algorithms that are implemented using a mixture of NVIDIA 's CUDA language and CUBLAS library. Given a computer with an NVIDIA GPU these functions may be substantially more efficient than native R routines.
- The rpud package (cran.r-project.org/web/packages/rpud/) provides an optimised distance metric for NVIDIA -based GPUs.
- The cudaBayesreg package (cran.r-project.org/web/packages/cudaBayesreg/) by da Silva implements the rhierLinearModel from the bayesm package (cran.r-project.org/web/packages/bayesm/) using NVIDIA 's CUDA language and tools to provide high-performance statistical analysis of fMRI voxels.
- The rgpu package (trac.nbic.nl/rgpu/) aims to speed up bioinformatics analysis by using the GPU.
- The magma package (cran.r-project.org/web/packages/magma/) provides an interface to the hybrid GPU/CPU library Magma (see below for link).
- The gcbd package (cran.r-project.org/web/packages/gcbd/) implements a benchmarking framework for BLAS and GPUs (using gputools - cran.r-project.org/web/packages/gputools/).
3.3.5. Summary

Using GPUs results in considerable speedups if algorithms fit to the single instruction-
multiple data paradigm (single instruction should be interpreted as sequential code). Typically, solutions are bespoken and need carefully crafted mappings to appropriate data structures. Big data sets need streaming solutions to overcome storage limitations but can lead to highly efficient, even real-time solutions.

3.4. Cluster Level Distribution

3.4.1. MapReduce

The distributed file system

In 2003, Google published a paper (Ghemawat et al., 2003) on their internal implementation of the so-called Google distributed file system. It is based on the idea of storing data on multiple machines with meta-data being available on a single node called namenode. Each block stored in GFS is automatically replicated to a configurable number of machines. On failure of one machine blocks are re-replicated automatically. Requests for data first contact the namenode for the file's location. Subsequent reads then directly contact the respective data nodes. This seemingly simple design led to a robust system that can survive multiple machine failures. Though a single point of failure the single namenode server did not prove as a major problem (it can be secured with standard measures such as RAIDed disks, redundant power supply etc.), however, it greatly simplified the implementation.

In 2003/4 some developers including Doug Cutting and Mike Cafarella were working on an open source internet scale search engine called nutch. Back then the system was built on top of Apache Lucene for indexing. Nutch would add crawling, document parsing, text extraction, link analysis and a fancy web frontend. Nowadays the project is focussing mostly on the crawling side and is re-using Apache Solr for configuration, Apache Tika for parsing.

When developing nutch the need for distributing storage and computation quickly arose. As a result, the nutch community adapted the patterns described in the original GFS paper developing a so-called nutch distributed filesystem, which later was migrated to a separate project called Hadoop distributed filesystem, or HDFS for short.

GFS itself only provides very basic distributed filesystem semantics that is far from what developers formerly working with regular databases are used to. To provide for a higher level data model, BigTable (Chang et al., 2006) was developed internally at Google. Basically BigTable provides a persistent distributed sorted map that allows for very efficient storage of sparse data. It supports versioned data entries. Based on GFS, it inherits its scalability properties. In the open source world with HBase and Hypertable there are equivalents readily available that are compatible with Hadoop. In addition databases for special tasks such as graph analysis have been developed (Malewicz et al., 2009), as well as hybrid solutions based on traditional database systems as well as Hadoop (Abouzeid et al., 2009). Extensions to BigTable (Peng, and Dabek, 2010) allow for incremental processing decreasing latency when compared to batch-only processing typical for computations running directly on GFS.

In recent years a huge ecosystem of projects surrounding Apache Hadoop has grown: With projects like Cascading, Hive or Pig it is possible to express complicated jobs consisting of chains of Map/Reduce processing steps without having to write a single line of Java code. With offers like Elastic Map Reduce from Amazon it is possible to run Apache Hadoop based Map/Reduce jobs in the Amazon compute cloud without the need to actually setup and
maintain a dedicated compute cluster. In addition companies like Cloudera, Rackspace, Karamsphere, Datameer and MapR provide services and products that built on top of Apache Hadoop, extend it or provide better performing implementations of several modules (e.g. the distributed file system) compared to the open source versions.

**Distributed computations**

After publishing details on GFS one year later Google revealed their approach to distributed programming (Dean and Ghemawat, 2004). It is based on the programming pattern Map/Reduce known from functional languages. When writing MapReduce programs the general idea is to do any data filtering during the map stage; aggregation and summarization is then done during the reduce phase. Map and reduce jobs are being distributed automatically by the framework. In this way the developer can focus on implementing the business logic without having to worry about networking issues, cluster layout or data distribution.

The canonical example for writing Map/Reduce programs involves counting the number of occurrences of patterns in some random text. When done on the Linux command line this is usually done by executing a command similar to the following:

```
cat $FILE | grep $PATTERN | sort | uniq -c
```

A similar structure would have to be implemented in the distributed setting as well. The grep would be implemented in the Map function, uniq in the Reduce function. Cat and sort would be provided by the framework itself and run on multiple machines. During Map the idea is to output any text matching the $PATTERN. In a real-world example this might be equal to some hostname found in a logfile or a number of user names of people interacting with the system. During the Reduce phase the framework guarantees that each call to the function sees all pairs individual patterns and their number of occurrence in each block of data as identified in the Map function. All that is left to do is to sum up and output the result.

The MapReduce framework guarantees data locality, that is Map functions are always executed close to those nodes holding the actual data to process, ideally on the same node. In addition on node failure the respective jobs running on that node are restarted on their equivalent replicas. In this way using the Map/Reduce framework makes it easy to write distributed, efficient and robust distributed programs.

In the open source world, there are quite a few Map/Reduce implementations with Hadoop being the most prominent one. An alternative is the Disco framework initially developed by Nokia written in Erlang. Alternative solutions are available in a number of databases including but not limited to Cassandra, Oracle, Vertica and others.

After introducing MapReduce for a wide number of problems, it quickly became apparent that there are various data analysis steps that are very common to many jobs. Functions like computing the average, min, max, filtering data, joining data are just a few examples of problems common to many data analysis pipelines. As a result with Sawzall (Pike et. All, 2005) a higher level language suitable for many data analysis steps was developed. In the mean time some of the sources have been published on Google Code.

The equivalent languages for Hadoop are Pig (Olston et al., 2008) as well as Hive. Pig originally was modelled after the Sawzall language for data filtering and aggregation. Operators available focus on extracting data based on patterns, aggregation and summarization. However due to growing demand developers have focused on adding SQL semantics to the language as well. On the contrary Hive comes from a different background: Developed at Facebook for use by data analysts it focused on SQL like queries from the start.
Hive makes it easy to migrate from standard database setups to Hadoop based installations as developers and analysts do not need to learn yet another language to interact with the system. Both higher level languages come with optimized MapReduce translations for the operations supported. They decrease development time for one-of analysis jobs dramatically. For recurring jobs it may still be beneficial to look into manually optimising jobs.

With Cascading there is a more Java centric workflow system available. With JAQL there is a JSON parsing language, initially developed by IBM and later on published, available that is capable of distributing jobs on Hadoop.

Internally at Google just recently libraries have been developed to allow for easier implementation of Map/Reduce jobs with Java (Chambers et al., 2010) as well as a system that allows for more interactive, less batch-oriented data analysis (Melnik et al., 2010). Melnik et al. (2010) focus on expressing data extraction jobs in a language developers are well familiar with, Java. Query plan optimization routines are used internally to translate this code into efficient MapReduce pipelines, potentially parallelizing or re-ordering whole operations for better performance.

**Alternative Map/Reduce Frameworks**

Apache Hadoop currently has the largest user base including users running it on small to medium sized clusters such as Zanox, nugg.ad as well as large corporate clusters at Yahoo!, Facebook (http://wiki.apache.org/hadoop/PoweredBy). As a result it is to be considered the most hardened distribution, tested in very diverse setups. Still it comes with a steep learning curve, both for developers as well as operations hosting an Apache Hadoop cluster.

![Twister programming model](http://www.iterativemapreduce.org/)

In addition to Apache Hadoop there are several vendors providing Map/Reduce integration for their products. Database providers such as Oracle support the definition of in-database Map/Reduce jobs (http://blogs.oracle.com/datawarehousing/2009/10/in-database_map-
reduce.html). On the other hand there are various providers of stand-alone Map/Reduce implementations such as Twister (http://www.iterativemapreduce.org/#home) and Disco (http://discoproject.org/).

Twister differs from Apache Hadoop and DryadLINQ in that it uses pub/sub messaging for all the communication/data transfer requirements which eliminates the overhead in transferring data via file systems (Figure 2). Essentially this boils down to trading some of Hadoop's failure resilience for better throughput and processing speed. The output <Key,Value> pairs produced during the map stage get transferred directly to the reduce stage and the output of the reduce stage get transferred directly to the combined stage via the pub-sub broker network. Currently Twister uses publish-subscribe messaging capabilities of NaradaBrokering (http://www.naradabrokering.org/) messaging infrastructure, but the framework is extensible to support any other publish-subscribe messaging infrastructure such as Active MQ (http://activemq.apache.org/).

3.4.2. DryadLINQ

DryadLINQ (http://research.microsoft.com/en-us/projects/dryadlinq/) is a compiler that translates LINQ programs to Dryad distributed computations where

- **Dryad** (http://research.microsoft.com/en-us/projects/dryad) is a general-purpose distributed execution engine for coarse-grain data-parallel applications running on Microsoft HPC networks. It extends the MapReduce paradigm in that a Dryad application can use an arbitrary graph structure for distributed computation. It combines computational “vertices” with communication “channels” to form a dataflow graph. It is designed as an infrastructure on which to layer simpler, higher-level programming models. Dryad handles job creation and management, resource management, job monitoring and visualization, fault tolerance, re-execution, scheduling, and accounting. A Dryad job is a graph generator that can synthesize any directed acyclic graph. A job is coordinated by the “job manager” that contains the application-specific code for constructing the job's communication graph along with code to schedule the work across the available resources. It is only responsible for control decisions and is not a bottleneck for any data transfers. The job manager runs either within the cluster or on a user’s workstation with network access to the cluster.

- **LINQ** (http://msdn.microsoft.com/en-us/netframework/aa904594) defines a set of general-purpose operators that allow applications to declaratively express query operations such as traversal, filtering, and projection within the .NET framework. It is an attempt to unify query languages such as SQL, Xpath, Xquery. It extends languages such as C# and F# with native language syntax for queries.

The general idea of dryadLINQ is that a programmer can use a high-level language and exploit cluster computing without having to know about Dryad and computer clusters in particular or about parallel or distributed computation in general. Features of LINQ are:

- **Declarative programming**: computations are expressed in a high-level language
- **Automatic parallelization**: from sequential declarative code highly parallel query plans (data flow graph) are generated.
- **Type safety**: distributed computations are statically type-checked.
- **Automatic serialization**: data transport mechanisms automatically handle all .Net object types.
- **Job graph optimisations**
  - Static: query optimization rules are applied to the query plan optimising locality and improving performance.
- Dynamic: run-time query plan optimizations automatically adapt the plan taking into account the statistics of the data set processed.

Reported applications (in: Y Yu, M Isard, D Fetterly, M Budiu, U Erlingsson, PK Gunda, J Currey. DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language, OSDI08 Proceedings of the 8th USENIX conference on Operating systems design and implementation, 2008) so far include:
  - Terasort benchmark (http://sortbenchmark.org/)
  - Pagerank
  - Clustering for detecting botnets
  - Statistical inference to automatically discover network-wide relationships between hosts and services on a network

Example (taken from: http://salsahpc.indiana.edu/tutorial/dryadlinq-intro.html#dryad). A word-count algorithm is implemented in style of a MapReduce application.

```csharp	namespace MapReduce
{
    public struct Pair
    {
        private string word;
        private int count;
        public Pair(string w, int c)
        {
            word = w;
            count = c;
        }
        public int Count { get { return count; } }
        public string Word { get { return word; } }
        public override string ToString()
        {
            return word + ":" + count.ToString();
        }
    }

    public class Program
    {
        static void Main(string[] args)
        {
            string uri = "file://\madrid-headnode\DryadData\hui\LineCount\input.pt";
            PartitionedTable<LineRecord> inputTable = PartitionedTable.Get<LineRecord>(uri);
            IQueryable<string> words = inputTable.SelectMany(x => x.line.Split(' '));
            IQueryable<IGrouping<string, string>> groups = words.GroupBy(x => x);
            IQueryable<Pair> counts = groups.Select(x => new Pair(x.Key, x.Count()));
            IQueryable<Pair> ordered = counts.OrderByDescending(x => x.Count());
            IQueryable<Pair> topPairs = ordered.Take(3); //top 3 most words
            foreach (Pair words in topPairs)
            {
                Console.WriteLine(words.ToString());
            }
            Console.ReadLine();
        }
    }
}
```
Code analysis:

- The Pair structure is used to store the <key, value> pair, where key is the “word” object, value is the “count” object.
- Applies the SelectMany operator to inputTable to transform the collection of lines into collection of words.
- The String.Split method converts the line into a collection of words.
- SelectMany concatenates the collections created by Split into a single IQueryable <string> collection named words, which represents all the words in the file.
- Performs the Map part of the operation by applying GroupBy to the words object. The GroupBy operation groups elements with the same key, which is defined by the selector delegate. This creates a higher order collection whose elements are groups. In this case, the delegate is an identity function, so the key is the word itself, and the operation creates a groups collection that consists of groups of identical words.
- Performs the Reduce part of the operation by applying Select to groups. This operation reduces the groups of words from Step 3 to an IQueryable <Pair> collection named counts that represents the unique words in the file and how many instances there are of each word. Each key value in groups represents a unique word, so Select creates one Pair object for each unique word. IGrouping.Count returns the number of items in the group, so each Pair object's Count member is set to the number of instances of the word.
- Applies OrderByDescending to counts. This operation sorts the input collection in descending order of frequency and creates an ordered collection named ordered.
- Applies Take to ordered to create an IQueryable <Pair> collection named topPair, which contains the 3 most common words in the input file and their frequency.

More information on Dryad Linq can be found at http://www.zdnet.com/blog/microsoft/microsoft-releases-dryad-concurrent-programming-code-to-academics/3385

3.4.3. Alternative Approaches

Hadoop and Dryag(LINQ) support particular architectural and design patterns and provide means to ease the task of programmers and algorithm designers. However,

- The patterns are restricted, though quite general in case of Dryad.
- Convenience of use is paid in terms of a certain amount of overhead.

Quite a number of data mining algorithm easily map to the Map-Reduce paradigm of Hadoop since they are inherently data-parallel such as k-nearest neighbour, Naïve Bayes, k-means, etc. Other algorithms that typically involve pruning are less suitable since pruning may result in global changes, which would imply an additional layer of exchange of information in a Map-Reduce system. Examples of this kind is Subgroup Discover (Grosskreutz and Rüping, 2009) or a distributed version of frequent item mining where transactions are distributed, item mining is executed locally, the frequency of itemsets are aggregated and, if the aggregation is complete (meaning the results of all nodes are available), communicated to all computation nodes to halt those computations involving itemsets the aggregated frequency of which are below the threshold. A blackboard architecture may be an alternative to support these kind of algorithms. A blackboard architecture is a distributed computing architecture where distributed applications share a common data structure called the “blackboard” and a scheduling/control process that organise updating and distribution of information, e.g., by broadcasting. A programming framework for this kind of architecture is for instance Oz-Mozart (http://www.mozart-oz.org/).
Programming frameworks cause some overhead in particular if high-level. For instance, examples demonstrate that the overhead of Hadoop file system has an impact for datasets that are not huge (Yong ong, 2010). More low-level bespoke solutions are an alternative. A favourite mechanism of this kind for cluster computing is MPI (Message Passing Interface), an API specification that allows processes to communicate with one another by sending and receiving messages, various implementations of which are available for many platforms as, e.g., OpenMPI, LA-MPI (http://public.lanl.gov/lampi/), LAM/MPI (http://www.lam- mpi.org/). OpenMPI (http://www.open-mpi.org/) is used in a number of papers on data mining (e.g. Yan et al., 2009, Woodsend et al., 2009, Fox et al., 2008, Corley et al., 2010, et al., Pješivac-Grbović 2007). MPI can be used for multicore and multiprocessor systems as well as for clusters. Compared with OpenMP, the difference is that OpenMP is thread-based while MPI is process-based. In combination with OpenMP and CUDA even clusters with nodes consisting of combinations of multicore CPUs and GPUs can be served (Pješivac-Grbović et al. 2007, Noaje et al., 2010). MPI interfaces are available for C/C++ and Fortran.

An alternative to MPI for the Java-world is Proactive (http://proactive.inria.fr/). Proactive is an open source Java library aiming to simplify the programming of multithreaded, parallel, and distributed applications for Grids, multi-cores, clusters, and data-centers. The main concept is that of activities being distributed, remotely accessible objects that interact through asynchronous method calls.

Furthermore, data mining algorithms available in R may be parallelised by mapping to clusters or Grids using R packages such as GridR (Wegener et al., 2008).

3.5. Higher level applications

3.5.1. Machine Learning Suites

With the tools described so far extraction and generation of basic as well as advanced statistics from log data is comparably easy. However when looking into more complex tasks such as grouping users by similarity, recommending items to users or classifying texts into categories these tools leave the developer with quite a bit of work to do. In the fields of data mining and machine learning a wealth of applications has been developed in past decades: there are algorithms for grouping items by their inter-similarity (so-called clustering), mining linked data, classifying items, coming up with recommendations based on past user behaviour and more.

A wide range of applications has been developed, both in the professional, research (http://jmlr.csail.mit.edu/mlmloss/) as well as the open source communities (http://mlloss.org/software/) dedicated to providing solutions to these kinds of problem settings. This chapter provides a brief overview of the applications available and compares them concerning their applicability for the Dicode project.

Professional providers such as Prudsys (http://www.prudsys.de/en/), SPSS Clementine (http://www.spss.com/de/), SAS (SAS Enterprise Miner Suite - http://www.sas.com/technologies/analytics/datamining/miner/) or TIBCO focus on providing complete data mining suites covering the whole pipeline of data import, preparation and machine learning. A more detailed list of data mining tool suites is available online at http://www.kdnuggets.com/software and http://www.kdnuggets.com/companies. Dicode's goal is to develop novel data mining and machine learning concepts. Naturally we expect tasks to be non-standard requiring innovative solutions. As such using closed systems that offer only limited extendibility would limit researcher collaborating on Dicode to an extent that is incompatible with the
project’s goals. From another perspective for most of these tools licensing costs are non-negligible—especially when taking into consideration that Dicode would re-use only a limited amount of the tools available.

Besides professional machine learning and data mining suites there is a large number of tool boxes available that have been developed during research projects (http://jmlr.csail.mit.edu/mloss/). Coming directly from academia the support for algorithms usually includes implementations of novel, cutting edge approaches that are still under discussion in the community. Support is readily available from the original authors, at least in cases where the software is still actively developed. However many packages can only be considered prototype level implementations that are far from ready for production use. They come in diverse licenses, partly proprietary, partly free. In most cases the diversity of file formats supported is huge and incompatible across tool boxes. This stands in stark contrast to the availability of well established, integrated machine learning suites to be explored in more detail below.

When looking for standard machine learning and data mining frameworks one of the most commonly used packages includes Weka (Hall et al., 2009). It comes with an easy to use graphical user interface that allows for experimentation, is extendable, and supports a wide range of data formats, analysis algorithms as well as quality evaluation modules. Originally developed at the University of Waikato, the framework is readily available under the open source license GPL, which allows for research as well as commercial use. A second option with a similar scope would be RapidMiner (http://rapid-i.com/content/view/181/190/), originally developed at the university of Dortmund the software currently is developed and supported by the spin-off RapidI. It is easy to use for beginners, comparably simple to extend. One major drawback of both suites is their weak scalability to large data sets. There is evidence that increasing the amount of data presented to the learning algorithm at hand can greatly improve its performance—even to a point where simple brute force algorithms can outperform more sophisticated ones simply due to their ability to process more samples (Banko and Brill, 2001). It is to be evaluated in the context of Dicode whether using elaborate sampling algorithms as a pre-processing step can help deal with that problem.

When looking into scalable learning frameworks two major approaches can be identified: One is based on distributing computation to several machines. This approach is mainly followed by some of the core algorithms provided for by Apache Mahout (http://www.ibm.com/developerworks/java/library/j-mahout/), (Owen et al., 2011). Several algorithms for building recommender systems, classification, clustering and frequent item set mining in Mahout have been implemented based on the MapReduce implementation of Hadoop. A major drawback of course comes from the added distribution overhead: Anecdotal evidence (http://tinyurl.com/5vkg8vhg) shows that below a comparably high threshold of available data the overhead of using a distributed implementation leads to decreased performance compared to a sequential implementation.

A second approach to improve performance is to reformulate algorithms not for parallelism but for one-pass, streaming processing of data. Mahout’s SGD implementation for classification follows this approach. So does the Vowpal Wabbit (http://hunch.net/~vw/) suite for fast online learning.

Of all machine learning frameworks discussed Apache Mahout currently has the most active development community. It has been reported to be used as part of the spam filtering pipeline at Yahoo!, for matching couples at Speeddate as well as a part of the recommendation modules at AOL and Foursquare. Mahout currently is shifting focus from being “Parallel based on Apache Hadoop” over to “Scalable to large datasets”. This includes
optimising existing implementations as well as refactoring to support for online, streaming
algorithms as was done with SGD already.

Due to the diverse nature of text mining tasks at Dicode it seems reasonable to assume that
one tool won't fit all machine learning challenges faced. As a result the need for using more
than one suite is to be expected.

3.5.2. Data preparation

No machine learning suite can operate without extracting meaningful features from the
objects to process. For text classification this may boil down to extracting relevant words,
phrases (so-called n-grams where n represents the number of words per phrase). Some of
this tooling is delivered with the frameworks themselves: Weka comes with simple feature
extraction support; Mahout relies on Lucene Analyzers for data pre-processing. However in
more sophisticated setups one may wish to annotate tokens with their part-of-speech tags,
normalize words such as “run”, “running”, “ran” to one single token, identify named entities
such as “New York” in contrast to “new building”.

Providers such as Terragram (http://www.teragram.com/) and Basis Technologies
(http://www.basistech.com/) have been proven to do very well for natural language
analysis in the past. Their tools are comparably easy to integrate into data pre-processing
pipelines. In more simple cases, OpenNLP (http://incubator.apache.org/opennlp) provides
basic natural language analysis as well. More involved text analysis is offered by the
OpenCalais (http://www.opencalais.com/) web service. Basically there are three options for
integration: Either the toolbox used features its own plug-in mechanism that allows for
integration of data pre-processing steps. A second way would be to use an external tool such
as UIMA (http://uima.apache.org) for data preparation. The third option is to facilitate
Lucene’s Analyzer architecture that allows for easy integration of external natural language
processing libraries.

When integrating data pre-processing frameworks there is a tradeoff to be made between
failure resilience and speed of processing: When forcing the framework to spill all results to
disk prior to future analysis this may unnecessarily slow down processing. On the other
hand building breaking points into the system avoids one component failing from affecting
the whole analysis pipeline.

3.6. Implications for Dicode

The design space for distributed data mining is huge and in parts unexplored. The advent of
multicores and GPUs opens new challenges in dealing with clusters of nodes which combine
multicore CPUs with GPUs. It seems that most of the data mining algorithms that have been
investigated are using data parallelism independent of the platform (Xiao, 2010). A question
not yet well understood is which platforms are most suitable with regard to a data
mining/machine learning algorithm and a data set. Experiments indicate that Map-Reduce
with an underlying file system is the platform of choice for data parallel algorithms if the
data set is really huge and somewhat persistent and if the data blocks are touched in a
streaming way, meaning that random access is avoided. Initial experiments have shown that
the overhead outweighs the gains if data fit into computer memory (and memory gets larger
all the time). Systematic analysis is needed here. More generally, memory and bandwidth of
communication seem to be more influential than computation speed. Although for highly
fine-grained parallel computations, the computational capacities of GPUs seem to be
extraordinary.
A tentative conclusion for further work in Dicode is that, depending on the envisioned scenarios derived from the use cases, the options within the design space should be systematically evaluated to provide the best and fastest (almost real-time) mechanism for decision making.
4. Data Mining towards Sense-making of Real-world Multi-faceted Data

4.1. Introduction

The previous section describes current research with respect to distribution and parallelisation of data mining algorithms. In practice however, the process of data mining exceeds the algorithmic point of view. Effective data mining depends on a blending of mathematically based techniques from statistics and computing, in additional to the algorithms, applied in the context of a specific problem setting. The choice of data mining approaches within Dicode will be driven by the requirements of the use cases. The subsequent sections only outline the general principles and problems which are commonly found in textbooks. However, specific application topics that are potentially relevant for the Dicode data mining process are presented in detail, such as text mining, subgroup detection, and business process management as well as in the field of scientific research process.

4.2. The Data Mining Process

Data Mining, also known as Knowledge Discovery in Databases (KDD), is a research field located at the intersection of Machine Learning, Statistics, and Database Theory, and is often defined as “the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data”. The characterization of data mining as a process is for example formalized in the CRISP Data Mining Process Model\(^3\) that defines an iterated process, in which previous decisions are frequently re-evaluated under the light of new insights as the process and hence the understanding of the structures behind the data increases. The CRISP process is shown in Figure 3 and consists of the following steps:

- **Business Understanding**: understanding the application domain. Identifying domain experts, understanding the problem-relevant domain-specific vocabulary, identification of important background knowledge.

- **Data Understanding**: understanding the data set that is being examined, i.e. its semantic, variable descriptions and specific data formats. Ontologies may be used for the formal description of data sets reducing overheads due to divergent data representation.

- **Data Preparation**: converting the data from the application-specific format into a format needed for the modelling step, cleaning the data, computation of derived features, feature and subset selection.

- **Modelling**: the actual analysis of the data using algorithms from Machine Learning or Statistics.

- **Evaluation**: checking the correctness and applicability of the model in the application context with respect to the goals of the analysis task.

- **Deployment**: integration of the discovered knowledge in the user domain. Practical application of the model, including pre-processing steps.

A similar structuring has been presented by SAS under the name SEMMA based on the operations that are available in the SAS Enterprise Miner software\(^4\). SEMMA consists of the

\(^3\) [http://www.crisp-dm.org](http://www.crisp-dm.org)

\(^4\) [http://www.sas.com/offices/europe/uk/technologies/analytics/datamining/miner/seemma.html](http://www.sas.com/offices/europe/uk/technologies/analytics/datamining/miner/seemma.html)
steps Sample, Explore, Modify, Model and Assess. These steps are covered by CRISP in the phases Data Preparation, Modeling and Evaluation.

The modelling step has been the focus of research in Machine Learning and Statistics. Many data analysis algorithms have been developed. Readily available open-source environments like R\(^5\), RapidMiner\(^6\) (formerly called “Yale”), or Weka\(^7\) contain a large, steadily growing variety of data mining methods. The other steps in the KDD process are usually treated in a more ad-hoc manner, even though it is widely acknowledged that these steps are very much responsible for the success of Knowledge Discovery projects (Pyle, 1999). For excellent overviews of the field of data mining in general, we refer the reader to the book written by Hastie, Tibshirani and Friedman (Hastie et al., 2002).

4.2.1. Characteristics of Data Mining

In the most recent literature, the fundamental monograph of Hastie, Tibshirani and Friedman (Hastie et al., 2002) explicitly identifies the new challenges in the areas of data storage, organization and searching, yielding to a new field of Data Mining where statistical problems exploded both in size and complexity.

David Hand (Hand, 2002) has given a suitable definition of Data Mining (DM). Hand defines Data Mining as the process for using non-trivial models to find out non-trivial and unsuspected relations in the data, which are useful and available. According to this definition, DM analysis applies suitable strategies based on the combined use of various statistical models. The goal is to discover both unsuspected relations among variables as well as dissimilarities among patterns that cannot be deduced in straightforward way from any query to databases. Finally, the terms ‘useful’ and ‘available’ qualify that the production of statistical information should involve effectively an added information value or profit for the end-user by improving the number of services, reducing the costs, increasing the business and discovering new impact factors. This is relevant not only in business management but also in other fields of interest such as in medicine and genetics. DM makes use of several statistical methodologies which are computationally very intensive, such as partitioning algorithms of segmentation and decision trees, selective algorithms for association rules.

\(^5\) http://www.R-project.org
\(^6\) http://www.rapidminer.com/
\(^7\) http://www.cs.waikato.ac.nz/~ml/weka/
discovery, as well as adaptive algorithms, re-sampling techniques and so on. The output result of these methodologies is often simplified by a comprehensive graphical description, aids to the interpretation and fruitful summaries.

A DM strategy can satisfy typical goals of Exploratory Data Analysis as well as Descriptive Modelling. The former aims to explore data along various directions and is normally performed when there is no prior information. Techniques of dimensionality data reduction are typically considered such as projection methods and factorial methods in combination with suitable visualization tools of analysis for large data sets. The latter provides a description of the data and of the data generating process. This can be obtained by estimating the empirical density distribution, by partitioning a multidimensional space in a given number of groups (i.e., cluster analysis and segmentation), by modelling the dependence relationships among the variables.

Another important goal can be achieved by a DM strategy, namely Discovering Patterns and Rules. In this case, DM employs algorithms for producing association rules (Agrawal et al., 1993) and methods of Pattern Recognition. The final aim of DM strategy is known as Retrieval by Content.

The researcher knows that the pattern and the objective of the analysis are to identify similar patterns in the data. This process yields textual data mining, in that occurrences of texts and images detection allow typologies to be specified as well as enabling the implementation of a searching procedure on a specific topic by keywords. Search engines on the Internet, such as Google.com, allow users to identify the web pages where the subject is coherent with the keywords specified by the user. Such research is done by suitable algorithms, e.g. PageRank (Brin et al., 1998), that assign a score to each document on the basis of the preference degree with respect to the subject of the research. Palms also adopt retrieval algorithms and technologies to recognize hand-made scripts.

Data Mining usually has to handle databases of the order of millions of records and, in some cases, of a gigabyte or a terabyte of units. In the ordinary life, several databases are continuously updated, such as those in the supermarket, in medicine, in telephone companies, in banks and insurances and so on. In the United States, one of the biggest retail distributors, the Wal-Mart chain, stores about 20 millions of transactions per day in the information system (Piatetsky-Shapiro, 1999).

When dealing with real world data sets, together with practical goals, it is necessary to take account also of the data quality, in particular the presence of missing data, the heterogeneity of the data collection and the coherence of various data flows. As a result, Data Mining cannot be simply viewed as an exploratory data analysis but it represents a new discipline of whose origin stands in the high dimensionality and in the non-standard typology of data, often summarized under the word complexity.

4.3. Machine learning and data mining technologies

With extensive overviews being available (e.g. (Bishop, 2006), (Witten et al., 2011), (Hastie et al., 2009)), this section only sketches some areas of machine learning and statistics that are relevant to the data mining problems and its challenges.

- **Data pre-processing:** before a data set can be processed by data mining techniques, several steps, such as the removal of obvious errors or the imputation of missing values, have to be performed. Provision of automatic operators to process the input sets into a form suitable for data mining forms the basis of all further data processing.
We refer to (Pyle, 1999) for an excellent overview of all issues regarding data preparation.

- **Unsupervised learning** is the task of extracting structures from data without any explicit information on which properties of the data are of interest. Unsupervised learning is used to structure data for further use and to allow for a quick overview of the data. Particular instances of unsupervised learning are:
  - **Clustering**: the records of a dataset can be partitioned into (disjoint, overlapping, or inclusive) subsets, depending on their properties. In context of data mining this is called “clustering”, and the subsets are called “clusters”. It is important to note that the partition is determined automatically, without human interaction, and only based on criteria that are defined on data properties (like a similarity measure on the dataset, for example). Some of the clustering algorithms additionally have parameters that indicate the number of subsets of the partition, or a similarity threshold that indirectly influences the number of clusters generated.
  - **Outlier Detection**: it is interesting to identify observations that do not fit to the bulk of the data, both because these observations may either be interesting by themselves (possible cases of fraud), and because the presence of outliers can have a negative impact on the performance of other data mining algorithms which benefit from the removal of outliers before the actual analysis.

- **Supervised Learning** is the task of learning to predict certain property of the data from labelled data, i.e. predicting the label of new observations from examples of observations with a known label. One approach to further structure supervised learning is to divide approaches based on the data type of the label (e.g. classification, regression, ranking). On the other hand, approaches can be structured based on the properties of the model that the produce.
  - **Logic-based learning** generated model in some hypothesis language based on a formal logic.
  - **Tree classifiers**: tree classifiers are a particular type of learning algorithms that represent models as trees where each inner node of the tree corresponds to a logical decision and each leaf represent a prediction of the label. According to experience, tree classifiers combine high-quality prediction performance with an intuitively understandable representation of the extracted model.
  - **Rule Learners** represent their model in the form of rules in some restricted logic. Similar to tree classifiers, rule learners are a particular type of learning algorithm with the advantage of providing understandable representation of the extracted model. The main difference between tree classifiers and rule learners is that, while the former yield a hierarchical, tree-based prediction model, the latter simply yields a flat list of rule or implications.
  - **Function based learning** generates models in the form of a numerical function and typically encode the learning problem as some kind of mathematical optimization problem.
  - **Support Vector Machines**: Support Vector Machines are a set of supervised learning techniques for learning a classification function. They are based on the idea to transform the input data in a high-dimensional space, and to separate the different classes by constructing a separating hyper plane in that space that maximizes the “margin” between the two data sets. Support Vector Machines have gained a lot of attention because they achieve very good classification results. On the downside, their model cannot easily be translated into a form that is easily accessible by humans.
• **Subgroup detection**: this method actively searches for patterns in the data with a significant deviation in the distribution of a target attribute and is specifically targeted at giving understandable representations of the found patterns. Using different quality criteria, subgroup detection can be targeted at finding local high quality patterns in the data, which complement the usual supervised learning setting that tries to find patterns that describe the data globally.

• **Other learning paradigms**
  - **Instance Selection and Active Learning**: decision making by human experts is usually instance-based, meaning that it is relatively easy for an expert to investigate a single document or a small set of documents while it is much harder to realistically assess the quality of a general rule. In conclusion, methods to select single informative instances from a database, for example such instances that are prototypical for a given pattern, are useful to make the inspection process more understandable and adequate to the way fraud experts work.
  - **Ontology-enhanced learning**: ontology is a formal description of real-world concepts. Ontologies can be exploited in data mining to guide the mining process and to transfer proven approaches and solutions to new data sets.
  - **Learning under concept drift**: concept drift is an important problem in the field of opinion mining.

• **Post-processing**: results of the data mining process, in particular the patterns that a learner has identified, may not be easily understandable to the user. In order to increase the understandability of the data mining approach, post-processing can be applied in order to simplify models or extract the most interesting rules from a large set of extracted rules.

• **Model optimization**: data mining algorithms usually possess one or more parameters that control the results in some way. The same is true for pre-processing operators, such as the selection of relevant features. As these parameters are usually only meaningful to a data mining expert, methods have to be implemented to automatically optimize the mining process, such that the end-user does not have to deal with technical details.

4.4. **Applications in Text Mining**

Different from data mining on databases, text mining (Feldman and Sanger, 2007) is applied to unstructured data. The regularities of language are very complex and include many exceptions to the rules. Traditional linguistics has identified the sub domains of syntax, semantics and pragmatics to describe the properties of natural languages. Current text mining theory and technology handles analysis and information extraction tasks from all three domains. To deal with these tasks, methods from computational linguistics (Manning and Schütze, 1999; Matsumoto and Mitkov, 2003), machine learning (Witten and Frank, 2005), and information retrieval (Manning et al., 2008) are used. An important strategy is to derive regularities from (large) collections of texts, rather than to design rule-based systems manually.

4.4.1. **Text indexing and relevance feedback**

Indexing algorithms are widely used in information retrieval (Manning et al., 2008) to enable search engines to efficiently find documents, based on terms or phrases provided by the
user. A recent example of a practical and efficient implementation is the Lucene (Gospodnetic and Hatcher, 2005) programming environment.

Relevance feedback adds a second iteration to the search: The user rates the documents that have been found according to his personal relevance. The set of most relevant documents is then used to determine the similarity of all documents in the database, and possibly to retrieve more relevant documents that have not been detected in the first trial. Originally, a vector space model was used to compute similarities, but today more sophisticated methods that are based on statistical algorithms improved the results (Manning et al., 2008).

4.4.2. Text annotation

The annotation of natural language texts with meta-data can be regarded as a pre-processing step which generates input data for subsequent processing steps. Parts of the text, like terms or phrases are assigned meta-information in order to clarify the semantic role of these text components. The most important ones are the following:

- Named entity recognition. Terms like proper names, geographical locations, enterprise names, or commercial products are identified and assigned their correct class label (like “name”, “location”, etc.). Simple methods for named entity recognition are based on pattern matching. For the English language these methods can be quite successful. But for languages with a richer morphology, the semantic role of terms can only be determined depending on the context. Recently several machine learning algorithms have been developed which can derive regularities of named entity use from collections of text examples. The most prominent one is the Conditional Random Field algorithm (Lafferty et al., 2001). FHG has been actively developing text annotation technology. (Paass and Kindermann, 2008) describes the technology of named entity recognition combined with active learning.

- Sentiment annotation. Texts usually not only express facts, but also personal views and emotions. These sentiments can be identified according to contextual regularities, much in the same way as named entities. Of course there are also sophisticated ways to express one’s sentiments between the lines. The more complicated cases of sentiment expression are not yet accessible to automated analysis.

- Resolution of co-references. Pronouns and other constructs are used in texts to refer to entities that have been mentioned before, or even entities which will be mentioned later. The algorithmic identification of these entities is a research goal, which has not yet been achieved fully, but is investigated heavily.

- Relation extraction. Semantic relations of the simple nature can also be identified from the typical usage of terms and syntactic constructs in texts. Examples of such relations are so called verb frames, which can be identified as syntactic structures, dominated by a verb and its direct and indirect objects.

4.4.3. Text classification

Text classification is a mapping of texts onto a system of predefined categories. The category system reflects properties of the texts, like topics (Bengel et al., 2004), text authorship (Diederich et al., 2003), semantic features of texts (informative, narrative, chat, etc.), and others. The category system can be unstructured, or it can contain an internal tree or graph structure. It can be applied to larger sections of text as well as single sentences. In the latter case, the classification usually applies to semantic features, and not to text topics. The most prominent classifier algorithms used for text classification are Bayes classifiers (Rennie et al., 2003), and the support vector machine (SVM) (Cristianini and Shawe-Taylor, 2000).
4.4.4. Text clustering

Clustering algorithms group data into subsets according to similarity relations. In our case of natural language expressions the similarity relation is computed on the basis of similarity measures on a vector space model (Manning et al. 2008), or co-occurrence relations. Text clustering algorithms like Latent Semantic Indexing - LSA (Deerwester et al., 1990), Probabilistic Latent Semantic Indexing - PLSA (Hofmann, 2001), or topic models created for example by Latent Dirichlet Allocation – LDA (Blei et al., 2003) can be used to define subsets of documents as well as subsets of the vocabulary, which is used in the texts. The latter feature allows one to represent documents for indexing operations by a reduced set of terms. Furthermore, terminological ontologies can be constructed from text collections using PLSA or LDA based systems, or term sets determined by these algorithms can be linked to existing ontologies (Spiliopoulos et al., 2007; Zavitsanos et al., 2007).

4.5. Subgroup Discovery

Subgroup Discovery is an instance of the learning task of local pattern detection, which is concerned with detecting patterns in data which are infrequent but very informative. This task is of particular interest for the Dicode project, since (i) it is not possible to scale up algorithms by the straightforward approach of sampling, i.e. always the complete data set has to be processed for pattern discovery, and (ii) supporting collaboration and decision support by data mining is more adequately solved by tools that can present different facets of a problem, instead of a global rule that explains everything (which would be adequate in automated decision taking).

A subgroup is a subset of a set of data, described by some rule, which has a statistically significant deviation in a selected variable of interest compared to the whole data set. For example, an interesting pattern within a subset of health care records may be found, or opinions within a particular time frame or area may deviate from the general average, where interestingness is made precise by the definition of a subgroup quality function.

The task of subgroup discovery is defined as to either find all subgroups with quality higher than a given threshold, or to find the k subgroups with the highest quality. Note that the completeness required in the subgroup detection task (“find all subgroups that...”) is the most interesting practical feature of subgroup discovery: while it is easy for users to check any single pattern for its statistical significance, such a manual approach does not solve the problem of how to find interesting groups in the first place.

Different subgroup patterns (e.g. for continuous or discrete target variables), search strategies and quality functions are described in (Klösgen, 1996) and (Klösgen, 2002). The search is arranged as an iterated general to specific, generate and test procedure. In each iteration, a number of parent subgroups are expanded in all possible ways, the resulting specialized subgroups are evaluated, and the subgroups are selected that are used as parent subgroups for the next iteration step, until a specified iteration depth is achieved or no further significant subgroup can be found. There is a natural partial ordering of subgroup descriptions. According to the partial ordering, a specialization of a subgroup either includes a further selector to any of the concepts of the description or introduces an additional link to a further table.

The statistical significance of a subgroup is evaluated by a quality function. As a standard quality function, the classical binomial test to verify if the target share is significantly different in a subgroup can be used. This z-score quality function based on comparing the target group share in the sub-group \(p\) with the share in its complementary subset balances.
four criteria: size of subgroup \((n)\), relative size of subgroup with respect to total population size \((N)\), difference of the target shares \((p - p_0)\), and the level of the target share in the total population \((p_0)\). The quality function is symmetric with respect to the complementary subgroup. It is equivalent to the \(\chi^2\)-test of dependence between subgroup \(S\) and target group \(T\), and the correlation coefficient for the (binary) subgroup and target group variables. For continuous target variables and the deviating mean pattern, the quality function is similar, using mean and variance instead of share \(p\) and binary case variance \(p_0(1-p_0)\).

Wrobel (1997) was the first to extend subgroup mining to multi-relational data. Knobbe et al. (2001) and Krogel et al. (2001) apply a static pre-processing step that transforms a multi-relational representation into a single table. Then, standard subgroup methods can be applied.

Subgroup discovery has been proven to be a successful method to produce an overview of the structures in large data sets and to identify possibly interesting local correlations and dependencies. The excellent understandability of subgroup rules, which are given in the form of simple if-then-statements, makes the output of a subgroup discoverer suitable even for users who are inexperienced with data mining techniques.

4.6. Data Mining and Business Process Management

In the Dicode approach, data mining is not seen as a solution by itself, but is relevant only in the context of solving the greater problem of decision making and collaboration support. It is hence important to view data mining as one step of a larger process and take care of the optimal integration in the larger context. Inspiration can be drawn from the field of business process management, which deals with standardized solutions for setting up and maintaining complex processes in the business world. In the business context, data mining is typically one step of a larger process. The data mining problem is well defined and the process is executed frequently. In this section, we present related work with respect to data mining in business processes.

In (Fayyad et al., 1996), the integration of data mining with other systems, e.g. database management systems or spreadsheet and visualization tools, is described as a research and application challenge. As applications today are getting more and more complex due to larger amounts of data or the distribution of data or resources, the problem of integration is also getting harder. Most important benefits of data mining for the business are achieved when the data mining results are deployed into a business process in a repeatable manner, which, e.g., involves the ability to rebuild, assess and apply models automatically. A survey on data mining techniques and business applications in the context of mining data from the business is given in (Hornick et al., 2006). There even exists some literature on integrating data mining in business processes from the end of ‘90s (Holsheimer, 1999).

In (Rupnik and Jaklic, 2009), a methodology for the implementation of data mining into operational business processes is proposed, consisting of the phases Exploratory Data Mining, Deployment of IT into the Business Process, and Operational Data Mining. The first phase consists of performing a CRISP-DM process and evaluating the readiness for an integration of the data mining solution. In the second phase, the data mining solution is deployed into the business environment, including changes in design and implementation of the business process (which means to perform BPR) and associated applications. The third phase describes the steps of the operational solution, including all CRISP-DM phases but business understanding and an additional phase for model updating.
Based on experience in software engineering, (Marbán et al., 2009) proposes a model for data mining engineering that includes engineering-related phases which are missing in CRISP-DM. They identify as open issue, that available process models specify what to do, but not how to do it. This is a point that we start addressing with this work. (Sharma and Osei-Bryson, 2009) also identifies the lack of guidance towards implementing particular tasks of data mining methodologies and introduces a framework for the implementation of the business understanding phase of data mining projects.

(Wegener and Rüping, 2010a) contributes an integrated role model for business, IT and data mining roles, an analysis on how the integration of data mining matches with business process reengineering best practices, and an approach for the evaluation of the integration of data mining processes into business processes. The work represents a further step towards an integrated environment which allows for flexible adding, exchanging or adapting data mining and business process tasks and fast experimenting, even on live systems, which is needed by modern data mining solutions in business applications.

Scheduling and workflow environments are slowly being addressed by business process environments (Hornick et al., 2006) with, e.g., frameworks like BPEL that allow and facilitate the inclusion of (external) web-services within business processes. JDM (Hornick et al., 2006) provides web service interfaces for such a kind of integration on API level that can be used in modern BPEL based frameworks. Java Data Mining (JDM) is a standard for developing data mining solutions. JDM provides a web service interface that can be used to set up, integrate and manage data mining processes in the context of BPMN and BPEL based environments. The authors present a use case on how to integrate data mining based on JDM into a business process based on BPEL in a Bank scenario. BPEL processes are designed which include calls to the JDM web services. These services interface with the JDM API on top of a Data Mining Engine. The authors use a role model with the groups Business Analysts, Data Analysts and IT.

In (Tsai and Tsai, 2005), a dynamic data mining process system is introduced. The idea is to set up each data mining activity as web service, to model and execute the data mining process in a BPEL environment and to get a PMML compliant model as result.

In (Russell et al., 2006), a set of workflow patterns describing the control-flow perspective of workflow systems is defined. Such process patterns have plenty of advantages (Atwood, 2006): BPM processes serve as both the specification and the source code. The modelled processes become the solutions deployed and provide a simple communication tool between end-users, business analysts, developers and the management. Process patterns provide a proven and simple technique to shorten the learning curve and improve productivity and quality of the processes designed, as they are simple to understand, learn and apply immediately.

Following this idea, (Wegener and Rüping, 2010b) presents a new process model for easy reuse and integration of data mining in different business processes. The approach is based on CRISP and includes the definition of data mining patterns, a definition of a hierarchy of tasks to guide the specialization of abstract patterns to concrete processes, and a meta-process for applying patterns to business processes. These data mining patterns allow for representing the reusable parts of a data mining process at different levels of generalization and provide a simple formal description for the reuse and integration of data mining. The approach was validated in a fraud detection case study in the health care domain.
4.7. Data Mining and Data Analysis Processes for Research

In recent years, developments in distributed architectures, such as grid technologies, have led to a vast increase in computing power and storage space that is available to end-users. The drawback of easily generating large volumes of data is that its analysis becomes increasingly hard and that even experienced researchers are facing problems to keep track of the state-of-the-art for a given analysis problem. Instead of being supported by information technology, the user is overwhelmed by an enormous volume of information. Hence, the user needs assistance in navigating in the space of appropriate KDD processes. The knowledge about solutions that were successfully used in the past for a similar problem provides a good base for building a supportive system for the user.

The work presented in (Punko et al., 2008) aims to develop an approach that assists the user without specific data mining experience to select a suitable KDD process. The fundamental idea that underlies this work is that data sets similar in content are likely to be analysed with the same KDD processes. This assumption allows reducing the problem of KDD process selection to searching for similar data sets. The main results of this work are the development of ontology-based characteristics to improve the dataset description, the development of a method allowing effective comparison of datasets based on the developed.

Workflow enacting systems are a popular technology in business and e-science alike to flexibly define and enact complex data processing tasks. A workflow is basically a description of the order in which a set of services have to be called with which input in order to solve a given task. Since the construction of a workflow for a specific task can become quite complex, efforts are currently underway to increase the re-use of workflows through the implementation of specialized workflow repositories. Driven by specific applications, a large collection of workflow systems has been prototyped such as Taverna (Oinn et al., 2004) or Triana (Taylor et al., 2006). As the high numbers of workflows can be generated and stored relatively easily it becomes increasingly hard to keep an overview about the available workflows. Workflow repositories and websites such as myexperiment.org tackle this problem by offering the research community the possibility to publish and exchange complete Workflows. An even higher amount of integration has been described in the idea of developing a Virtual Research Environment (VRE (Fraser, 2005)). Due to the complexity of managing a large repository of workflows, data mining approaches are needed to support the user in making good use of the knowledge that is encoded in these Workflows. In order to improve the flexibility of a workflow system, a number of data mining tasks can be defined:

Workflow recommendation: Compute a ranking of the available workflows with respect to their degree of interest to the user for a given task. As it is hard to formally model the user's task and his interest in a workflow, one can also define the task of finding a measure of similarity on workflows. Given a (partial) workflow for the task the user is interested in, the most similar workflows are then recommended to the user.

Metadata extraction: Given a workflow (and possibly partial metadata), infer the metadata that describes the workflow best. As most approaches for searching and organizing workflows are based on descriptive metadata, this task can be seen as the automation of the extraction of workflow semantics.

Pattern extraction: Given a set of workflows, extract a set of sub-patterns that are characteristic for this workflow. A practical purpose of these patterns is to serve as building blocks for new workflows. In particular, given several sets of workflows, one can also define the task of extracting the most discriminative patterns, i.e. patterns that are characteristic for one group but not the others.
Workflow construction: Given a description of the task, automatically construct a workflow solving the task from scratch.

Since workflow systems are getting more complicated, many researchers have addressed the development of effective discovery techniques, particularly for this field, during the last years. Public repositories that enable sharing of workflows are widely used both in business and scientific communities. While first steps toward supporting the user have been made, there is still a need to improve the effectiveness of discovery methods and support the user in navigating the space of available workflows. A detailed overview of different approaches for workflow discovery is given in (Goderis, 2008). Most approaches are based on simple search functionalities and consider a workflow as an atomic entity. Searching over workflow annotation like titles, textual description, or discovery on the basis of user profiles belongs to basic capabilities of repositories such as myExperiment (de Roure et al., 2008), BioWep, Kepler or commercial systems like Infosense and Pipeline Pilot.

In (de Roure et al., 2008), a detailed study about current practices in workflow sharing, re-using and retrieval is presented. To summarize, the need to take into account structural properties of workflows in the retrieval process was underlined by several users. Authors demonstrate that existing techniques are not sufficient and there is still a need for effective discovery tools.

Apart from workflow sharing and retrieval, the design of new workflows is an immense challenge to users of workflow systems. It is both time-consuming and error-prone, as there is a great diversity of choices regarding services, parameters, and their interconnections. It requires the researcher to have specific knowledge in both his research area and in the use of the workflow system. Consequently, it is preferable for a researcher to not start from scratch, but to receive assistance in the creation of a new workflow.

Designing new workflows by reusing and re-purposing previous workflows or workflows patterns has the following advantages:

- Reduction of workflow authoring time.
- Improved quality through shared workflow development.
- Improved experimental provenance through reuse of established and validated workflows.
- Avoidance of workflow redundancy.

4.8. Implications for Dicode

It is important to recognize that data mining in real-world applications is one step towards a greater goal. While data mining is well suited to extract knowledge of large amounts of data with a more or less homogeneous structure, making sense of possibly very heterogeneous data requires human understanding and guidance.

While scalable high-performance data mining algorithms (as discussed in Section 3) are the key technology to enable data understanding in data-intensive scenarios such as those investigated in the Dicode project, the practical success of these data mining solutions depends on external factors such as: (i) the ability to integrate the data mining solutions into the application environment, (ii) enabling the user and domain expert to guide and control the data mining process and include their domain knowledge, and (iii) the overall usability of the data mining system, in particular the ability to re-use existing solutions and built upon proven solutions.
In the context of Dicode, this bears particular importance with respect to the integration of data mining and collaboration support tools into one integrated system. A common understanding of the roles of users and the role of support tools such as data mining is necessary to ensure an optimal combination of the respective strength of human understanding, decision making, and automated data mining.
5. Collaboration and Decision Making Support

5.1. Introduction

Collaboration and decision making support technologies play an important role in Dicode. More specifically, these technologies aim to:

- establish the semantics that shape the overall context in which the Data Mining services operate, taking into account that initial settings are important for the efficient operation of Data Mining services;
- offer a collaborative environment that allows users “immerse” in Web 2.0 interaction paradigms and exploit its enormous potential to collaborate through reviewing, commenting on and extending the shared content (i.e. data retrieved from the Data Mining Services);
- maintain chains of views and opinions, accompanied by the supporting data (as they result from the Data Mining services), which may (i) reflect, at any time, the current collective knowledge on the issue under consideration, and (ii) justify a particular decision made or action taken;
- achieve group sense-making, by exploiting the outcomes of Data Mining services in argumentative discourses that reflect the diverse stakeholders’ views.

The term “collaboration support software” refers to software that is designed to support a group of people involved in a common task to achieve their goals (http://en.wikipedia.org/wiki/Collaborative_software). The emergence of the so called Web 2.0 era introduced a plethora of collaboration tools which provide engagement at a massive scale and feature novel paradigms. These tools cover a broad spectrum of needs ranging from knowledge exchanging, sharing and tagging, to social networking, group authoring, mind mapping and discussing. For instance, Delicious (http://delicious.com) and CiteULike (http://www.citeulike.com) provide services for storing, sharing and discovering of user generated Web bookmarks and academic publications, respectively. A different set of applications focuses on building online communities of people who share interests and activities (social networking applications). Facebook (http://www.facebook.com), MySpace (http://www.myspace.com) and LinkedIn (http://www.linkedin.com) are representative examples of this category. Another set of Web 2.0 tools aims to collectively organize, visualize and structure concepts via maps to aid brainstorming and problem solving. Their particular emphasis is on the visual appearance of the workspace. Tools such as Thinkature (http://www.thinkature.com) and FreeMind (http://freemind.sourceforge.net) fall into this category. Systems such as online discussion forums, Debatepedia (http://wiki.idebate.org) and Cohere (http://cohere.open.ac.uk/) aim at supporting online discussions over the Web. phpBB (http://www.phpbb.com) and bbPress (http://www.bbpress.org) are Web 2.0 applications enabling the exchange of opinions, focusing especially on providing an environment in which users can express their thoughts without paying much attention to the structure of the discussion.

Generally speaking, there are many different categories of Web 2.0 collaboration and decision making support tools. Classified upon their basic purpose, the most popular categories of them are:

- **Mind mapping**: tools in this category permit the representation of ideas and concepts that can be connected to form diagrams. They emphasize the visualization and structure of ideas aiming at studying and organizing information to achieve problem solving and...
decision making. Tools of this category include Compendium, Thinkature, FreeMind and XMind.

- **File sharing**: this category contains tools that enable the distribution and facilitate access of digital information in the form of files. Files are usually maintained centrally although distributed approaches are also very popular. Representative tools of this category are Dropbox, Google Docs and Box.net.

- **Collaborative editing**: tools in this category permit the joint authoring of documents via individual contributions. Representative tools are Wikis, Content Management Tools (CMS) and Google Docs.

- **Social networking**: Social networking tools permit the building of social networks or social relationships through which users may share interests and activities. Representative tools of this category are Facebook, LinkedIn and Twitter.

- **Note taking and annotation**: Tools in this category enable the creation of shared comments, notes, explanations, or other types of remarks that can be attached to any part of a resource (such as document or Web page). Applications of this category include Annotea, OntoMat and SHOE Knowledge Annotator.

- **Project and task management**: This category features tools that enable the coordination of projects by managing all related resources and enforcing all necessary constraints. Tools in this category include Basecamp, ActiveCollab and Redmine.

- **Argumentative collaboration**: This category enables the creation of argumentative discourses where a group of people exchange positions and arguments in order to achieve consensus on the issue discussed. Representative tools of this category include Araucaria, DebateGraph, Compendium, Hermes (Karacapilidis and Papadias, 2001), and Cohere.

In the next sections, we first review the above categories more thoroughly in an attempt to identify state-of-the-art functionalities and solutions offered by representative tools, while also considering the requirements of the Dicode project. Then, adopting a community perspective, we review community modelling, monitoring and adaptive collaboration support approaches.

### 5.2. Mind mapping

Mind mapping tools enable the creation and edit of so-called “mind maps”. A mind map is a diagram used to represent words, ideas, tasks, or other items linked to and arranged around a central keyword or idea (Figure 4). Mind maps are mainly used to generate, visualize, structure, and classify ideas, and act as an aid to studying and organizing information, solving problems, facilitating sense-making, making decisions, and writing (http://en.wikipedia.org/wiki/Mind_map).

Mind mapping tools are similar to note-taking tools but differ from them in the way the information is structured and on their emphasis on using elaborate visual cues (shapes, colors, etc) to represent concepts and their relationships.

A mind map could be seen as a depiction/overview of a certain piece of knowledge. It is a collection of “topics”, formed around a central idea (could be a single word or a whole phrase) which forms the central topic. On the central idea, in a radial way, associated ideas/concepts are added. More formally, a mind map includes (Buzan, 1989): (i) The central topic/idea which is unique for each mind map; it is the point where the map starts. (ii) Main topics stemming from the central idea; each main topic is connected to the central idea through an associate line. (iii) Main topics may be further analysed to subtopics; subtopics...
are of lesser importance. The central topic, main topics and subtopics form a connected graph.

![Figure 4: An example of a mind map (Source: http://artificialeyes.net/files/adjusting-to-eye-loss-mind-map-770.jpg)](image)

Representative tools of this category are:

**MindMeister** ([http://www.mindmeister.com/](http://www.mindmeister.com/)) is an online tool basically used for brainstorming by using mind maps. It enables the creation of maps by creating the central idea of the issue under consideration. Users can work on maps synchronously and in real-time and permits exporting maps in various formats. Offline work mode is also available with the use of a desktop version which permits uploading of maps to the online site. An API allows third party applications to access and edit available maps. In order to enhance collaboration in data intensive environments with multiple online users, notifications via emails or SMS are supported making users aware of changes on shared maps. Users may also use the focus (zoom in/out) feature to browse and work on maps with a large number of topics. In such a case, the feature of expanding/collapsing the subtopics of a topic helps in having an overall view of the map. A filtering feature may be used to isolate part of the map by using specific criteria which include the name of the user having edited part of the map, the text of an idea on a map or the predefined icons (actually reflecting tags) that may have been added on an idea. History views of maps are also supported where the entire history since the creation of a mind map is stored as different versions.

**Mindomo** ([http://www.mindomo.com/](http://www.mindomo.com/)) has been designed to support map creation and enhance collaboration in business, home and education. Midomo maps (diagrams) are in the format of a tree, starting from a single point, which may be a word or a whole sentence. The starting point branches out and divides until a whole tree has been created. A Mindomo map has a central topic which is then ‘decomposed’ in other topics or subtopics. The Mindomo user is able to interconnect to topics or subtopics by using the relation tools or group map.
items with the “Boundary” tool. A desktop version, enables working on desktop in offline mode and publishing maps on the online account. To enhance collaboration in data intensive situations, Mindomo, supports collapsing/expand option along with hiding and showing of user selected parts of the map. “Tagging” is also possible by using a number of specific icons (“tags”) which may be added to a topic’s visualization giving more information about its status such as its importance. Zooming in/out is also available to make browsing on large maps easier. The history of a mind map provides a list of all the actions that have been performed on the specific map. The filtering feature may be used to isolate part of the map by using various criteria. A search tool allows the identification of topic and subtopics whose content match the specified text.

Bubbl.us (http://www.bubbl.us/) is a mind mapping tool, similar to Mindmeister, designed to enhance brainstorming. With Bubbl.us, a user is able to create online maps and share them with other users. A mind map created with this mapping tool may be embedded in a blog or web site, can be saved as image or sent by email. Bubbl.us map consists of bubbles. Features such as bubble text, size, and color are user defined. Two bubbles may be connected with a directional line (connection). An “Undo” feature enables users to trace all steps that have been taken to create a map. Maps can be exported in various formats such as XML, html or as images. To cope with information overload, Bubbl.us provides a zoom in/out tool which may be efficiently used to help browsing and scrolling in maps that include many bubbles. Coediting of maps is also possible resulting in sharing maps among the members of a group of users.

XMind’s mind map (http://www.xmind.net/) follows the classic format of mind maps created by other popular mind mapping systems: the main idea is a root in the center of the map and branches stem from the central root. To support business domain tasks, XMind provides features to enhance project management and milestones timelines. Different visualizations are possible via fishbone (Ishikawa) diagrams, tree charts and organization charts. Boundaries, relationships, markers, labels, notes, audio notes, hyperlinks and graphics can be added to a mind map. Files can also be added either as floating topics or as attachments to topics functioning as subtopics. Exporting of maps is also possible in a variety of formats. Online collaboration is supported by specifying access rights of individual users. To deal with data intensive maps, XMind supports a filtering mechanism which allows users to make a specific part of the map visible by selecting specific attributes (markers or labels). Topics that fulfill the filtering criteria imposed are highlighted while the topics not fulfilling the criteria are faded out. The extending/collapsing feature of the subtopics of a topic may also be useful when dealing with maps containing a large number of topics. Boundaries are used as a topic aggregation mechanism to which info concerning the state of the topic can be attached.

5.3. File sharing and collaborative editing

File sharing tools refer to online tools which, at least provide a repository where each user may upload his files. Usually, uploaded files may be shared with other users, regularly backed up and restored at the user’s will. Many file sharing platforms integrate online office suits that may be used to create, edit or view some types of the uploaded documents. Based on their emphasis on sharing documents, they are also used as collaborative editing environments – with some of them providing advanced services for such support - and for this reason such tools are also included in this category.

Collaborative editing tools permit the joint authoring of documents via individual contributions. Wikis are the most representative systems in this category. Wikis allow users
to freely create and edit Web page content using any Web browser. They support a simple markup language with which new pages and crosslinks between internal pages can be created (Leuf and Cunningham, 2001). They have been used to support a great variety of tasks which include creating collaborative workspaces, managing shared knowledge and personal note taking. In general though, Wikis have been used as a collaboration platform to jointly author documents that reflect the group’s understanding with respect to the issue under consideration. Hence, the development of Wiki content is driven by the community.

Representative tools of these two categories are:

*DropBox* (http://www.dropbox.com/) is a web-based service enabling users to store and share files and folders on web folders located on the DropBox server. DropBox provides strong integration with the operating system and permits the creation of local DropBox directories the contents of which are synchronized with the directory allocated for the user on the DropBox server. The contents of the web DropBox directory may be accessed from any computer or devices (such as cell phones) equipped with the DropBox client application. Sharing of files with other users is based on an invitation model. To deal with information overload issues, especially in cases where many users collaborate and coedit files, DropBox provides version history and file recovery. An online list provides access to all events that have taken place (such as file editing, time editing took place, name of the editor) in a shared folder. Notifications inform users about changes on their web folders (e.g. when a shared file is changed by a user having access to the file). Files can be hierarchically organized via subfolders.

*Humyo.com* (http://www.humyo.com/) is an online storage service appropriate for sharing and synchronizing files across different computers. Humyo provides a Web interface for uploading and downloading files. File sharing is possible via an invitation model. Sharing is also possible via social networking sites such as Facebook and Twitter. Additional services include: file backup and restore, file synchronization with desktop folders, SSL encryption and on-the-fly file editing via the Zoho office suite. To deal with data intensiveness, Humyo provides online teamspaces. In teamspaces, files can be shared within teams emphasizing on secure storage spaces and exchange of large files. A filtering mechanism allows retrieval of files according to their content type (such as images, audio and video files). Searching of files is also available by specifying criteria that include the file name, file size, file folder location as well as file creation and last modification date.

*Box.net* (http://www.box.net/) is a cloud-based content management system. Box.net’s core services emphasize the sharing of folders and files via URLs. Sharing can be achieved by simply providing the email of users with which files and folders should be shared. Tagging of files-folders is also available though the web interface. Box.net provides a number of features such as full text search, synchronization with desktop folders, project collaboration and workflow management, advanced administration and security control, integration with third party services (e.g. Zoho, Google Analytics, Twitter) and access of files through third party sites (e.g. Google Apps, LinkedIn). In addition, the Box.net offers also features designed to enhance collaboration in data intensive environments such as document versioning, file and folder tagging and filtering mechanisms.

*Google Docs* (http://docs.google.com/) is Google’s online office suit. A user of Google Docs may either upload files on the web folders of his account or create/edit documents of various formats that include spreadsheets, html forms, presentations and drawings. Google Docs may be used as storage space and files may be downloaded and edited on the user’s computer. History revision, searching, sorting based on file type and tagging are provided for all saved documents, features which may be useful especially in data intensive environments. To aid document creation, Google Docs provides a wide range of document
templates. Google Docs support user collaboration: each user can share his documents with others Google users or collaborate to edit/create a document. A link to the shared document is created and may be sent to other users either by email or by using Buzz, Facebook or Twitter.

**MediaWiki** (http://www.mediawiki.org/wiki/MediaWiki) is web-based software system which enables the collaborative authoring of web pages. It is one of the most widespread wiki software and has been used support a great number of popular wikis (Barrett and Daniel, 2008) which include Wikipedia, Wiktionary, Wikiquote, Wikibooks, Wikispecies, Wikinews, Conservapedia (http://www.conservapedia.com/) and Metapedia (http://en.metapedia.org/wiki/Wikipedia). MediaWiki provides rich authoring possibilities by enabling amongst others embedding of mathematical formulas and the attachment of files. Each wiki pages is associated with a discussion forum in order to enable collaboration of page authors in case a consensus must be reached. MediaWiki offers customization in multiple layers and allows extending existing functionalities via Extensions (http://www.mediawiki.org/wiki/Extension_Matrix). Extensions are usually developed by third party developers by using the MediaWiki API. In order to address data intensive issues and information overload, MediaWiki provides features such as watchlists which enable the notification on changes to pages the user is interested in, page history and versioning which is accompanied by services that allow the comparison between versions and the visualization of changes as well as mechanisms to control access to pages. Wiki pages can be hierarchically organized into categories. A keyword-based search mechanism permits retrieval of wiki pages which contain the specified keywords.

**Confluence** (http://www.atlassian.com/software/confluence/) is wiki software which is also commercially available. It enables the creation of Spaces, that permit grouping of content items. Pages can be created by anyone and can be organized in hierarchies (“families” of pages) offering an effective page categorization. A blog is associated to each page. Tagging (or labeling) a page, commenting on a page or blog entry, retrieving older versions of the page, page exporting (Word, PDF) and access control are some other available page features. Confluence embeds a rich text editor for creating pages. Editing may be also done by using Microsoft Word (the document can be saved back in Confluence). Editor features include image importing (thumbnails/galleries), content embedding (Powerpoints, Word Documents, Excel files, PDFs, multimedia content). A number of macros allow adding extra functionality on the content of a page. Searching in Confluence equals to searching in all spaces (page content), space descriptions, and content of the attachments (as long as they are of certain type) by applying a number of criteria (Where/What/When/Who). Confluence includes a number of functionalities to minimize complexity and perform in data intensive cases; In particular, it supports notifications, separate spaces, access control mechanism, page categorization and tagging, searching and notification mechanism are useful features to avoid information overloading.

**PBworks** (http://pbworks.com/), formerly known as PBWiki, is a wiki software emphasizing in particular on business, education and personal usage. A PBworks workspace contains a wiki and Pages/files section where files may be uploaded. Workspaces may be shared with other users. The creator of the wiki is able to set the access level of each user; different access rights may be granted to users. PBWorks plugins are available to add multimedia content (images, video and slideshows) on the wiki. Attaching files to pages is possible, and a great range of formats are supported. Backup services are also provided. To cope with data intensive situations, PBWiki offers a number of features. Wiki versioning is provided meaning that the administrator of the wiki is able to track all changes in the wiki’s content since its creation. Notifications are sent via email (users may select the frequency of
notifications) whenever the wiki’s content is changed. Each page of the wiki may be tagged with user selected tags and the whole workspace may be exported to a zip file. A keyword-based search mechanism is available for locating content/pages of the pages of the workspace and tags may be also used in searching.

5.4. Social networking

Social networking refers to the creation of social structures by connecting individuals with various types of ties such as friendship, kinship and common interests. Although a social network is possible to be established with personal contact (especially when small social groups are engaged), online social networking has become very popular with the development of social networking websites. Members of such websites form an online community and they are able to socialize with other users.

Social networks may be focused on specific group interest (e.g. Academia.edu (http://academia.edu/) focuses on academics researchers) while others (e.g. Facebook) don’t. In most networks, membership is usually open meaning that every user may become member no matter what his hobbies, beliefs or interests are. Such networks support the creation of subnetworks in them, allowing, at the end of the day, the formation of groups sharing common interests.

Social networks can be analyzed in order to reveal specific properties. Social network analysis (Hanneman and Riddle, 2005, Wasserman and Faust, 1994) focuses on the mapping and measuring of relationships and flows between people, groups, organizations, computers, URLs, and other connected information/knowledge entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes. Social network analysis provides both a visual and a mathematical analysis of human relationships.

Representative tools of this category are:

Facebook (http://www.facebook.com/) is an application to build large-scale social networks. Users in Facebook create profiles which consist of personal information photos, personal interests and contact information. Profiles can be updated and can be public or private. Information, ranging from simple text messages to entire videos, can be shared between users. To deal with the enormous number of its members and the vast amount of information contributed, Facebook incorporates the News Feed feature. News Feed is a summary of recent actions of Facebook members who are in some way related to a specific user (may be his “friends” or “friends” of his “friends”). Information included on News Feed has been previously marked as public. Apart from the News Feed feature that notifies users about actions that have recently taken place, notifications by email or SMS are also available (as long as the user has selected it from his personal profile). User may also create a Facebook groups which consists of people sharing common interests.

MySpace (http://www.myspace.com/) used to be the most popular social network until the advent of Facebook. Creating a profile in MySpace enables user to create profile pages (not only text pages as in Facebook but also html pages) that may be used for uploading and sharing photographs, links, videos and music. Profile pages contain also personal user information such as the two standard blurbs available (“About Me” and “Who I ’d like to meet”), “Interests” and “Details” section. It is up to the user which of this information is shared with other MySpace users. On October 27th 2010, MySpace released a beta version and they have announced that they no longer compete Facebook as a general social network site but, instead, MySpace, from that time on, will be music-oriented and their main target will
be younger audience. Similarly to Facebook, in order to cope with information overloading from the large community of users, MySpace incorporates MySpace Groups. The creator of a MySpace group may accept or reject a user’s request for participation in the group. A space user has the option to get notifications by email upon other users posting comments or uploading content on his profile page. Searching for people or content on MySpace is also possible through the provided search mechanism.

LinkedIn (http://www.linkedin.com) is as social network basically aiming at professionals and business market. Unlike other popular social networks, its main focus is to keep in contact colleagues, alumnus and in general people who share common professional interests. Users in LinkedIn create profile pages listing professional information such as studies and former/current jobs. Users (and in particular companies) may search for people with specific professional profile when opting to hire a new person. They can also advertise a job position in order to “invite” LinkedIn users to apply for it. Connections between users can be established whereby forming networks. To deal with data intensiveness, LinkedIn has implemented the feature of LinkedIn groups. Each user is also able to create events and automatically propagate the event to all of his connections. A notification mechanism makes users aware of recent events in their networks (such as new connections).

Twitter (http://www.twitter.com/) is a website that allows users to send and read short (up to 140 characters) messages called tweets. Its focus is on supporting “social grooming” or “peripheral awareness” i.e. making people aware of what the people around them are thinking, doing and feeling even when co-presence is not viable8. Users may subscribe to receive other user’s tweets – a process known as following and the subscribers are called followers. “Following” is the main mechanism of twitter through which social networks are created. To overcome the problems originating from the immense amount of messages exchanged between users, Twitter displays messages in reverse chronological order and enables organizing messages via hashtags. Hashtags – words in messages prefixed with the character # - allow categorization of messages and function as links that display all messages belonging in that category. In addition, many third party tools are available to organize and filter messages that include Twitter Times (http://tweetedtimes.com/), Cadmus (http://thecadmus.com/) and Greasemonkey scripts.

5.5. Note taking and annotation

Note taking is about recording the most critical information out of a larger amount of information. The source of this information may be a lecture in a class, a project meeting or an everyday scheduling of a person’s activities. There are various techniques/methods one may use to keep notes. More specifically, (i) Charting: A graph with symbols or a table is created. (ii) Outlining: The most general information begins at the left while more specific group of facts indented with spaces on the right. (iii) Mapping: Mind maps for note taking include important information as nodes placed on a tree structure. Nodes are connected with lines reflecting a relationship. (iv) Cornell method: The basic information is recorded, spaces are left between the basic information points. Gaps are filled with extra information when the recording of the most important information has finished.

Electronic note taking has to do with software tools that have been developed to enhance the process of note taking. Notes, apart from text content, may include files, multimedia content and worksheets. A user’s notebook may be either personal or be shared with other online users through which collaboration can be achieved.

Representative tools of this category are:

**Zoho Notebook** ([http://notebook.zoho.com/](http://notebook.zoho.com/)) supports the creation of notebooks that may contain various types of content. A Zoho Notebook user may own multiple notebooks, each one containing a number of different pages (user may add one or more blank, text, sheet or web page in each notebook). Each notebook page is filled with content (text, images, audio, video, RSS feeds, documents, URLs, files, sheet) that can be either created online by using the online provided tools or uploaded from an external source (website or local computer). Zoho Notebook provides Web-browser plug-ins to enable instant web clipping. A drawing tool enables adding shapes onto the content to signify annotated places. Pieces of content within a notebook can be shared among users. Shared content can be jointly edited. To support collaboration in data intensive environments, Zoho NoteBook groups may be used to allow sharing content among specific users. Versioning of shared notebook content allows keeping track of changes and modifications.

**Evernote** ([http://www.evernote.com/](http://www.evernote.com/)) is an online tool designed to support note taking. A user is able to store permanently notes which may include text, images, audio files and handwritten “ink” notes. Evernote is also available as a desktop application. The Evernote user has the option to create notebooks on his desktop and synchronize them with his online account. Evernote provides also plugins for a number of Web-browsers. To cope with data intensive cases, EverNote Notes are organized in notebooks (being essentially “folders” of notes). Each Note may be tagged and organized in folders. A Search mechanism is also available for spotting a desired note.

**SimpleNote** ([http://simplenoteapp.com/](http://simplenoteapp.com/)) is a note taking application used for online storage of notes, lists and ideas. A SimpeNote note consists of a title and text. SimpleNote supports versioning. Changes on user’s notes are tracked and the user is able, though a slider bar, to go back in time and get previous versions of his notes. Apart from the web space allocated to each user for taking his notes, a number of clients are provided (mainly by third party developers) to synchronize the online notes with the user’s personal computer (windows, mac os x) or mobile devices (iPhone, Android, Palm webOS). SimpleNote is an open platform and many extensions, scripts, and plugins are being built to provide extra functionality. To enhance note management in accounts with a large number of notes, a searching mechanism has been developed to allow text search in the notes’ text. A note may have one or more tags that may be used either in searching or in listing. A note may be also pinned to rank notes based on their importance than the others and subsequently be moved on the top of the notes’ listing.

### 5.6. Project and task management

Project and task management tools aim at supporting the overall process of managing all relevant resources towards the achievement of predetermined project goals and objectives. In particular, such tools attempt to help in achieving the project goals while enforcing and obeying to preconceived project constraints.

In general, project and task management software enables an integrated approach to manage planning, scheduling, monitoring, budgeting, resource allocation of large projects in an attempt to overcome the problem of using different software for each abovementioned process. Typically, project management tools permit in addition content management, provide notifications via email and feature process related awareness mechanisms.

Project and task management tools provide also collaborative service to enable cooperation and coordination of virtual teams, i.e. teams that are geographically distributed.
Collaborative aspects of these tools come in the form of sophisticated user, group and role management which enable the assignment of responsibilities to individual and restrict their access, sharing of documents, forum-like discussion boards to provide a central point for project related discussions as well as notifications when specific events occur.

Representative (open source) tools of this category are:

**Basecamp** ([http://basecamphq.com/](http://basecamphq.com/)) is a Web-based project management system implemented in Ruby, that focuses mainly on making project management tasks easy to use. Basecamp provides milestone and deadline management, task-lists, wiki style content management, file sharing, time-tracking and messaging systems. Although Basecamp supports different users, it does not allow setting up different permissions on a per user basis. In general, all users have access to all project related resources (such as tasks, milestones etc). With respect to data-intensiveness issues, Basecamp attempt to alleviate the effects of massive email exchanges which are frequent during project management tasks, by offering the “Message Board”. The “Message Board” permits keeping project related messages and discussions centrally and accessible to all.

**ActiveCollab** ([http://www.activecollab.com/](http://www.activecollab.com/)) is a closed-source Web-based project management system. ActiveCollab provides milestone management, tickets, checklists, time tracking, file and content management and per project discussion boards. ActiveCollab supports also invoicing. Tickets in ActiveCollab are project tasks which are assigned to one or more project participants (assignees). Tickets can be organized into categories and can be associated with milestones. Tickets can also be broken up into smaller sub-tasks each of which has a due date, priority and a list of assignees. Tickets that have been completed are automatically archived. “Checklists” constitute an aggregation of tasks which are completed when every task they contain is completed. ActiveCollab supports permissions (in the form of roles), enabling the specification of resources and functions users can have access to. Dealing with data-intensive issues, ActiveCollab provides features such as awareness mechanisms, filters and milestone “zoom-in”. In particular, with respect to awareness mechanisms, ActiveCollab features ‘Reminders’ which permit remind participants about a ticket, a discussion or a page. The recipient of such reminders is configurable: reminders can be send to individuals, task assignees and people who commented on a specific issue. Regarding filters, ActiveCollab enables the filtering of task assignments based on various criteria that include date, priority, assignees and project. Milestone “zoom-in” permits the access to milestone details which include only all related discussions, files and tasks.

**Redmine** ([http://www.redmine.org/](http://www.redmine.org/)) is an open-source Web-based project management system based on the Ruby-on-Rails framework. It provides an issue management system, flexible role based access control model, Gantt charts and calendars, time tracking, file sharing and content management system (Wiki), feeds and email notifications, per project discussion forums and integration with SCM (SVN, Git, CVS, Mercurial). The access to project resources is controlled via permissions that can be organized into roles. In Redmine, issues have statuses that indicate their stage. Statuses in Redmine are user-defined to meet the specific project needs. Workflow transitions can be set up based on issue type and user role. To manage the complexity and projects, Redmine enables the creation of sub-projects. Redmine also automates some functionalities as for example in the case of Gantt charts and calendars which are calculated based on the start and due time of issues. Awareness services are also available by publishing project activities, news, issues, and issue changes either as email or as Atom feeds.
5.7. Argumentative collaboration

Argumentation is a verbal activity, which is usually conducted in an ordinary language. A speaker or writer, engaged in argumentation, uses certain words and sentences to state, question or deny something, to respond to statements, questions or denials and so on (van Eemeren et al., 1996). Argumentation is also a social activity which, in principle, is directed to other people and is directly connected to reaching conclusions through logical reasoning. It always relates to a particular opinion about a specific subject and the need for argumentation arises when opinions concerning this subject differ.

Over the past years, several argumentation models have been proposed to address specific needs. These models can be classified in three main categories (Bentahar et al., 2010):

- **Monological models.** These are based on informal languages and mainly focus on the structure of the argument. For these models, the relationships between the different components of an argument are of crucial importance (rather than the relationships between the different arguments). Models that fall into this category are Toulmin’s model (Toulmin, 1958), the argumentation model proposed by Reed and Walton (Reed and Walton, 2003) and the model by Anscombe and Ducrot (Anscombe, 1995, Anscombe and Ducrot, 1983).

- **Dialogical models.** Unlike monological models, dialogical models consider the macro-structure of the arguments and highlight the argumentation process in a dialogue structure. They focus on fallacies arguments (arguments that seem to be valid but actually are invalid). Representative models of this category are those proposed in (Hamblin, 1970), (MacKenzie, 1979), (MacKenzie, 1981), (Amgoud et al., 2000a), (Amgoud et al., 2000b), and (Bentahar and Labban, 2009).

- **Rhetorical models.** The models of this category consider neither the external nor the internal structure of the arguments, but the rhetorical structure of the argument. They are based on the audience’s perception of the argument and evaluative judgment and do not focus on establishing the truth of a proposition. Examples of such models are reported in (Grasso, 2002) and (Grasso, 2003).

Argumentation (support) systems are software systems designed to help people take part in various types of dialogues in which arguments are exchanged. Such systems have been used in domains such as commerce, education, law and planning (Scheuer et al., 2010). Generally speaking, the design of software systems that can adequately address users’ needs to express, share, interpret and reason about knowledge during an argumentative collaboration session has been a major research and development activity for more than twenty years. Technologies supporting argumentative collaboration usually provide the means for discussion structuring and visualization, sharing of documents, and user administration. They support argumentative collaboration at various levels and have been tested through diverse user groups and contexts. Furthermore, they aim at exploring argumentation as a means to establish a common ground between diverse stakeholders, to understand positions on issues, to surface assumptions and criteria, and to collectively construct consensus.

Representative tools of this category are:

*Araucaria* (http://araucaria.computing.dundee.ac.uk/doku.php) is a representative tool of this category, which enables argument analysis through diagrams. Araucaria arguments are being built by selecting phrases from a user defined text. Each selected text corresponds to a node on the argument diagram and lines (relationships) can be dragged from one node (premise) to another (conclusion). A user may also add missing premises on the diagram, create/modify/load/save an argumentation scheme or delete components.
The analyzed arguments are saved on the disk in a portable format (AML-Argument Markup Language based on XML format). Araucaria has been designed to enhance teaching and critical thinking (Rowe et al., 2006).

DebateGraph (http://debategraph.org/) is another interesting approach. A DebateGraph map includes two main components, “ideas” and “relationships” between two of them. Several mechanisms have been implemented to support large scale argumentation and collaboration in data intensive environments. For instance, there is a “history” mechanism allowing one to browse the other users’ actions. In addition, a progressive visualization of the argumentation map is supported, in that only a part of the map is shown on the user’s screen (one “idea” and the “ideas” that are directly connected to it). A user may traverse the map by moving one level at a time. Awareness mechanisms are also available; a user may be informed for changes on the map either by email or by RSS. A search mechanism is also available to look for text in user’s private maps and maps publicly shared. Finally, different views of a map are available, namely “zoom” (for zooming in the selected area of the map), “tree” (hierarchical displaying), “explorer” (including a zoom in/out tool), “macro” and “hub” (includes a discussion area for each map).

Compendium (http://compendium.open.ac.uk/) is another software tool designed for mapping information, ideas and arguments. Ideas on a Compendium map are expressed by using different types of nodes which are linked together with different types of relationships. To deal with data intensive environments, Compendium includes a number of features. More specifically, multiple level maps are supported (a map may include another map). There is also a zoom in/out tool to help in focusing and browsing maps with a large number of nodes. The “aerial” view of a Compendium map is also helpful in large maps. A search mechanism has been also implemented and criteria such as a node’s tag, the author’s name, the node’s date/text are used for searching. In order to reposition the nodes on a map, a user can multiple-drag nodes (nodes are selected by creating a rectangle containing them and are dragged together). Finally, a user is able to store node “bookmarks” for easily locating a node on a large map.

CoPe_it! (http://copeit.cti.gr/) is a Web 2.0 tool designed to enhance collaboration by sharing opinions and resources in communities of practice. A CoPe_it! user can create a personal or collaborative workspace, join and contribute to an existing workspace and add/share content through a workspace. Various argumentation items may be uploaded and linked. Users can collaborate in either an asynchronous or a synchronous way. A scalable approach has been followed in CoPe_it! to support the diversity of users’ requests; features and functionalities range from human understandable to machine understandable (multiple content projections are used). CoPe_it! supports a number of features to enhance collaboration in data intensive cases. For instance, the “minimap” of a workspace provides an overview of its contents. Also, there is a “review/history” mechanism, through which one may follow the evolution of a workspace. In addition, multiple items may be grouped together. Finally, a filtering mechanism enables one to view argumentation items fulfilling specific criteria, such as the item’s title, date, author and type.

Cohere (http://cohere.open.ac.uk/) is another online visual tool used to create, connect and share ideas (which may appear in private, public or group spaces). Cohere mimics the most popular social sites as it encapsulates the idea of people, people pages and groups. Ideas created by a user are gathered in the user’s personal web page (a form of personal web notebook of ideas). The Cohere notebook includes the user created websites, their annotated clips, the connections created between the ideas and the list of people/groups that the user shares content with (De Liddo et al., 2006; Shum, 2008). To deal with data intensive situations, Cohere offers various filtering mechanisms (e.g. ideas and their connections can
be filtered according to their type). Tagging of ideas is also available, while a search mechanism may exploit the ideas’ text, the associated tags, the name of a user or group and the text appearing on a connection.

5.8. Collaboration tools overview and implications for Dicode

In order to devise a roadmap towards the identification of the appropriate collaboration technologies and functionalities to be further exploited in the context of Dicode, we categorize the tools discussed in the previous sections according to several dimensions. In particular, we classify the categories and individual tools according to: (i) their collaboration objective (i.e. what the tools aim to achieve via collaboration?), (ii) functionalities that individual tools provide, (iii) the cognitive overload issues they are prone to, and (iv) the countermeasures that these tools introduce to overcome cognitive overload issues. These dimensions are important for the Dicode project, as they outline the solution space not only in terms of objectives and functionalities, but also in terms of technologies to remedy cognitive overload and data-intensiveness issues.

Table 1 classifies the presented categories of collaboration tools according to the objective they mainly aim for. We distinguish the following classes: (i) community building, when tools aim for locating people and building ties between them, thus forming communities, (ii) communication, when the objective is simply to enable peer-to-peer communication between participants, (iii) coordination, where the objective is to align the actions of a group of people to achieve a common goal, (iv) sense making, when tools aim at the exchange of opinions and ideas and generate meaning of the exchanged items, and (v) decision making, where the aim is to select a course of action among several alternatives.

<table>
<thead>
<tr>
<th></th>
<th>Community building</th>
<th>Communication</th>
<th>Coordination</th>
<th>Sense making</th>
<th>Decision making</th>
</tr>
</thead>
<tbody>
<tr>
<td>MindMaps</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Collaborative editing</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Social Networking</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note taking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Project/task management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Argumentative collaboration</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Table 1: Collaboration tools categories and their objectives.

Next, we classify each tool according to the services and functionalities it provides. Table 2 illustrates this classification⁹. The importance of this table is to show that although tools aim at specific objectives, this is not necessarily associated with specific services and functionalities, and that these tools provide a wide range of collaboration services which are not typical for their kind. The set of services and functionalities that are examined has been

⁹ Due to its simplicity and limited collaboration support, we do not include Twitter in the following discussion.
selected based on their potential usefulness for the Dicode project and include: (i) *discussion*: the ability to facilitate exchange of ideas and opinions between groups of people (brainstorming) and make these ideas and opinions subject to comments, (ii) *archiving*: the ability to archive and organize the items under collaboration or the entire collaboration, (iii) *visualization*: the ability to provide advanced visualizations of the collaboration space such as graphs, (iv) *annotation*: the ability to annotate or tag resources in the collaboration space, (v) *chat*: the ability to enable peer-to-peer real-time interaction between participants, (vi) *awareness*: the ability to inform participants on the actions of other participants, (vii) *task lists*: the ability to maintain list of tasks along with computational support to evaluate and enforce these lists, (viii) *file sharing*: the ability to share files (documents and other resources) between participants of the collaboration, (ix) *document management*: to manage (organize) documents available in the collaboration space, and (x) *user & role management*: the ability to define users and roles through which the levels of access to the resources are controlled.

<table>
<thead>
<tr>
<th></th>
<th>Discussion</th>
<th>Archiving</th>
<th>Visualization</th>
<th>Annotation</th>
<th>Chat</th>
<th>Awareness</th>
<th>Task lists</th>
<th>File sharing</th>
<th>Document management</th>
<th>User &amp; role management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mind mapping</td>
<td>1, 2, 4</td>
<td>1, 2, 3, 4</td>
<td>2, 4</td>
<td>2</td>
<td>1, 2</td>
<td>2</td>
<td>1, 2</td>
<td>1, 2</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>Collaborative editing</td>
<td>9, 10</td>
<td>7, 8</td>
<td>5, 6</td>
<td>5, 6</td>
<td>12, 13, 15</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
</tr>
<tr>
<td>Social networking</td>
<td>12, 13</td>
<td>12, 13</td>
<td>12</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
</tr>
<tr>
<td>Note taking and annotation tools</td>
<td>15, 16</td>
<td>15, 17</td>
<td>15</td>
<td>15</td>
<td>15, 16</td>
<td>15, 16</td>
<td>15, 16</td>
<td>15, 16</td>
<td>15, 16</td>
<td>15, 16</td>
</tr>
<tr>
<td>Project/task management</td>
<td>19</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>19, 20</td>
<td>19, 20</td>
</tr>
</tbody>
</table>

**Table 2**: Collaboration tools and their functionalities.


We then analyze collaboration tools categories according to the sources of cognitive overload (Table 3) and the countermeasures taken (Table 4). By the term ‘source of cognitive overload’ we refer to the characteristics of information that may lead to cognitive overload situations in each tool, while by ‘countermeasures’ the solutions that the tools make available to remedy cognitive overload. The analysis is based on the extensive list of causes and countermeasures, which are reported in (Eppler and Mengis, 2004) and are relevant to the Dicode project. More specifically, we identify the following sources of cognitive overload that need to be addressed by collaboration services in Dicode: (i) *rising number of information*:
the information items brought into collaboration increase as the collaboration proceeds; such increase may not be gradual but may appear in bursts, (ii) uncertainty of information: the inability to assess quickly the relevance of the available information, (iii) information diversity and increasing number of alternatives: the situation in which diverse types of information exist and the number of solutions increases as the collaboration proceeds, (iv) ambiguity of information: the situation where information can be interpreted in several ways, (v) complexity of information: the degree of interrelationships of information, (vi) intensity of information: the importance of particular information items, (vii) increase of information dimensions: the situation in which the way the available information brought in during collaboration can be combined with an increasing number of other items or can be considered along different aspects and dimensions, (viii) information quality and value: the degree of worth of information and (ix) overabundance of irrelevant information: the excessive amount of irrelevant information which leads to a low signal/noise ratio of the items in the collaboration space.

<table>
<thead>
<tr>
<th></th>
<th>Rising number of information items</th>
<th>Uncertainty of information</th>
<th>Info diversity &amp; increasing no of alternatives</th>
<th>Ambiguity of information</th>
<th>Complexity of information</th>
<th>Intensity of information</th>
<th>Increasing dimension of information</th>
<th>Information quality value</th>
<th>Overabundance of irrelevant information</th>
</tr>
</thead>
<tbody>
<tr>
<td>MindMaps</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Collaborative editing</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Social Networking</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Note taking</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Project/task management</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Argumentative collaboration</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Table 3: Causes of information overload for each category of collaboration tools.

From the above analysis, it results that a plethora of collaboration technologies is available, each of them aiming to support different objectives. Analyzing the tools with respect to the services they provide, it is evident that although tools belonging in the same category provide a common core set of services, they supplement them with services which are not typical for their category. While elaborating on the issue of sources of cognitive overload in each category of collaboration tools, it also revealed that all categories are prone to such concerns. However, the analysis also shows that each tool attempts to address the related data intensiveness and cognitive overload issues by introducing particular services or approaches, which aim at alleviating the severe consequences. In the same line, each category of tools favours particular cognitive overload countermeasures, which are explicitly designed to address the problems that occur in a particular collaboration context. When each tool is used independently, the available countermeasures may provide the required support to address information overload issues. However, when an integrated approach of the presented tools must be considered, i.e. when two or more tools have to be deployed to address collaboration needs, the countermeasures may be insufficient and of limited use.
This is mainly because the countermeasures of each tool have a particular scope which is derived from the collaboration objective. Hence, tools which belong to different categories but exhibit common countermeasures conceive them in different terms, thus raising concerns on how to consider them when these tools have to be used jointly.

<table>
<thead>
<tr>
<th></th>
<th>Structuring information (interlinking, aggregating, annotating)</th>
<th>Visualization (graphs)</th>
<th>Formalization</th>
<th>Simplicity</th>
<th>Customization &amp; personalization</th>
<th>Levels of detail/summaries</th>
<th>Awareness (notification, history, versioning)</th>
<th>Search &amp; filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mind mapping</strong></td>
<td>1, 2, 3, 4</td>
<td>1, 2, 3, 4</td>
<td>1, 2, 3, 4</td>
<td>1, 2, 3</td>
<td></td>
<td>1, 2, 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Collaborative editing</strong></td>
<td>5, 6, 7, 8, 9, 10, 11</td>
<td>5, 6, 7, 9, 11</td>
<td>9, 10</td>
<td>5</td>
<td>5, 7, 8, 9, 10, 11</td>
<td>6, 7, 8, 9, 10, 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Social networking</strong></td>
<td></td>
<td>12</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td>12, 13, 14</td>
<td></td>
</tr>
<tr>
<td><strong>Note taking and annotation tools</strong></td>
<td>16, 17</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td>16, 17</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Project/task management</strong></td>
<td>20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td>18, 19, 20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4**: Countermeasures taken by collaboration tools.


In Dicode, signals are strong that such an integrated approach to collaboration is required. In particular, in the project’s context, argumentative collaboration, collaborative editing, note taking and mind mapping tools look promising to address the foreseen collaboration needs. Yet, these tools must be considered in an integrated manner. This integrated consideration of diverse collaboration tools raises questions on how to redefine the available countermeasures and adapt them in this new environment. In Dicode, we envisage collaboration tools grafted with effective cognitive overload countermeasures, which do not limit their focus to particular collaboration objectives but provide their services in situations where heterogeneous collaboration tools must interoperate.

### 5.9. Community Monitoring and Adaptive Collaboration Support

Virtual communities, where people with common interests and goals work together sharing experience, constructing collective knowledge, and taking collective decisions, are playing a vital role in the modern work practices in both business and academia. Stepping on this social phenomenon, Dicode will capitalise on the collective knowledge accumulated in
virtual communities, as well as provide intelligent support to facilitate collaboration and decision making in such communities.

In a broad sense, virtual communities vary from loosely structured to closely-knit ones. An open, loosely structured community involves a large number of people with diverse interests, membership control is generally not imposed, and there are no restrictions of the interaction with the community information space. Examples of this kind of communities, such as forums or web-blog communities, are widely available on the web. Participation in such communities is on a voluntary basis. They are highly dynamic, include a broad range of participants, joining and disconnecting at any time. Research has shown that such communities are subjected to power-law distributions, and consists of overlapping clusters which can evolve over time (Palla et al, 2005; Palla et al, 2007). Loosely structured communities become informal drivers forming trends and broader influences.

In contracts, closely-knit communities involve a smaller number of people and usually exist in relatively well defined organisational or educational settings. This kind of communities are characterised with a common goal (e.g. planning experiments or making judgement on a patient’s condition), shared interests among all members, some commitment to participate in collaboration activities, high level of interaction, and active participation, which sometimes involves pre-assigned roles. Closely-knit communities usually involve well-formed teams, e.g. people working on a research project or collaborative medical diagnostic teams assessing patients’ conditions. These communities have controlled membership for accessing the community’s space and resources, and are closed for the outside world. They have well-defined norms, responsibilities, and work in established trust and reputation models. Although tightly focused, closely-knit communities may be influenced by broader trends and developments in large, loosely coupled communities.

The community monitoring and adaptive community mechanism in Dicode will concern closely-knit communities. The monitoring mechanism will focus on the collaboration and decision making process within the closely-knit community, while the support mechanism can include support for internal interaction, collaboration, and decision making processes, as well as support from external, large, loosely structured communities.

5.9.1. Approaches for Community Modelling

Recent research trends look at intelligent ways to support the effective functioning of online communities. In this line, personalisation and adaptation techniques play a crucial role. The effectiveness of personalised support provided to virtual communities depend on what is known about a particular community and in which areas the community may need support.

Modelling virtual communities has recently become very popular in different research areas. In user modelling, modelling group of members provides the grounds for generating group recommendations (Masthoff, 2004). In social networks, community modelling aids the discovery of relationships between people and among communities (Lin et al., 2008). We review approaches from both user modelling and social network below.

5.9.2. Discovering connections

A fairly simple and elegant community model is presented in (Cheng and Vassileva, 2006). It is based on a list of topics based on the resources VC members are sharing. A reward factor is calculated to measure the relevance of each contributed resource to the current topic the VC is working on. Each member has an individual user model consisting of the reputation measure of that member in the VC (Cheng and Vassileva, 2006). An earlier work in the same group presented a more elaborate relationship model (Bretzke and Vassileva, 2003). Users’ interests are modelled in (Bretzke and Vassileva, 2003) based on how frequently and how
recently users have searched for a specific area from the ACM taxonomy, and user relationships are derived based on any successful download or service that took place between two users. A more recent approach by Kleanthous and Dimitrova (2008, 2010) employs the metadata of the resources shared in the community along with an ontology representing the community context, and derives a semantically relevant list of interests for every user.

5.9.3. Modelling interests

User interests have been extensively studied. For example, (Davies et al., 2003) presents an approach where user interests are extracted as keywords from the user profiles and other web content shared by a user in the community. An ontology is then accessed, where associations are derived with ontology concepts and further recommendations are made to users. Interests are also used in finding relationships between users or connections in social graphs. (Li et al., 2008) and (Kleanthous and Dimitrova, 2008) extract interests based on tags users ascribe to items posted online. Relationships/associations between users are derived based on their tags. Members can then be connected by interest similarity between them. A different approach is followed in (Tian et al., 2001), where the community model represents the interaction activities that happen in the VC. User interests are modelled according to the interactions each user is performing in the VC and associated to the core lexicon of the VC. Shared interests or relationships are also modelled based on social interaction activities of users, and are linked to the VC lexicon. (Kleanthous and Dimitrova, 2008; 2010) also model user interests based on resources members are uploading or downloading. However, this exploits semantic enrichment of the uploading/downloading activities by using, in addition to the resource key words, concepts extracted from an ontology. This uses semantically-enriched data to extract interest similarity between community members.

5.9.4. Modelling expertise

Interests of users are usually associated with expertise, especially in social network research (Song et al., 2005; Fu et al., 2007; Lin et al., 2007; Zhang et al., 2007). (Zhang et al., 2007) extracts shared interests in a discussion based on posting/replying threads. Based on the discussion topics a member of the community is contributing to, his interests and expertise are extracted; subsequently, user interest relationships are obtained. (Fu et al., 2007) is following a similar method but is mining email communication networks. Relationships are inferred according to the expertise/interests of members, which are extracted from communication recorder in their email conversations. Modelling expertise relations plotted as graphs is also explored in (Song et al., 2005). A relational network is extracted according to people’s publications. The expertise/interests of a person are obtained by his previous publications; and two people are considered related if they have publications in the same research area. Relevant to expertise is a person’s influence in the community. This can be derived applying social network formulas based on the community graph (e.g. see (Kleanthous and Dimitrova, 2010)).

5.9.5. Community graph models

Recent research employed graph theory to model communities and relationships between members (Hubscher and Puntambekar, 2004; Kay et al., 2006) or members’ interactions in general (Falkowski et al., 2007; Falkowski and Spiliopoulou, 2007). In (Hubscher and Puntambekar, 2004), the individual user model represents the conceptual understanding of a user, based on which a graph network is constructed. Similarities are then extracted according to a user’s conceptual understanding, and group models are derived based on the distance between members in a graph. (Kay et al., 2006) uses the notion of interaction
network to represent relationships between users in a learning community. Two members are related if they have modified the same resource; hence, they appear connected in the interaction graph. (Falkowski et al., 2007) considers the exchange of messages as interaction between two users, represented in a graph. A relationship between two users exists if they have engaged in some message exchange (Falkowski and Spiliopoulou, 2007). (Kleanthous and Dimitrova, 2009; 2010) develop community graph based on semantic relationships, in addition to the interactions between users - an edge connecting two members represents their semantic similarity to each other, and the relevance of this link to the community’s domain.

5.9.6. Community patterns

The community model can be analysed to automatically detect problematic cases which can be used to decide when and how interventions to the community can be done, offering support to improve the knowledge sharing processes in the community (Kleanthous and Dimitrova, 2009). Research in organisational psychology identifies processes that can have an impact on collaborative processes, and are important for the effective functioning of teams and closely-knit communities (Mohammed and Dumville, 2001; Ilgen et al., 2005). Previous research has focused on three key processes: Transactive Memory - members are aware how their knowledge relates to the knowledge of the others (Wegner, 1986); Shared Mental Models - members develop a shared understanding of what the common goal is and how each one is contributing to this goal (Mohammed and Dumville, 2001); and Cognitive Centrality - members who hold strong relevant expertise are influential; members of effective communities gradually move from being peripheral to becoming more central and engaged in the community (Ilgen et al., 2005). It has been shown that community patterns based on these processes can be derived from the community graph (Kleanthous and Dimitrova, 2009; 2010).

Interaction activities (e.g. communication and argumentation) are crucial for collaboration and decision making. Hence, in Dicode, we envisage a novel mechanism for modelling relationships between content and people based on interaction data – both interaction with content (e.g. a medical image) or argumentative discussions between members. Input for the community modelling mechanism can be interaction log data, including provenance data of collaboration and decision making activities, as well as engagement in dialogue/argumentative interactions. We also expect that certain patterns, related to collaboration and decision making, will be detected by analysing the community interaction data. Finally, Dicode can take advantage of the ontology that can be used to relate people and content.

5.9.7. Approaches for community support

There is a growing interest in providing intelligent support for teams, groups and communities.

Visualization techniques are among the most popular methods that can be employed to present group and community models in a graphical way, to help groups function more effectively (Kay et al., 2006; Upton and Kay, 2009), to motivate community participation (Cheng and Vassileva, 2006), and to make members aware of reciprocal relationships (Sankaranarayanan and Vassileva, 2009). The key limitation of visualization techniques is their passive influence on the functioning of the community, e.g. while examining graphical representations members may not be able to see how their contribution could be beneficial for the community as a whole and what activities they can engage in.
Different tools and algorithms have been developed to support people in locating expertise on a specific subject inside groups or VCs (Shami et al., 2007; Zhang et al., 2007). There is a growing body of research on intelligent group/community interventions, e.g. notification (Ardissono et al., 2009), feedback (Baghaei and Mitrovic, 2007), or promotion of cognitively central members (Bretzke and Vassileva, 2003; Farzan et al., 2009). Community interventions aiming at improving the functioning of the community as an entity are presented in (Kleanthous, 2010). This includes pointing at connections between members which have not been exploited or encouraging cognitively central and peripheral members to engage in interactions beneficial for the whole community.

5.9.8. Implications for Dicode

Dicode will require a new approach to support collaborative teams. On the one hand, teams can be supported to better conduct internal activities linked to collaboration and decision making (e.g. consider all possible aspects when making a judgement, compare opinions from diverse sources). On the other hand, teams can be made aware of external processes related to their decision making process, such as trends (e.g. specific data sources can be used for specific purposes), influences (e.g. patients' attitude to a drug may be influenced by the overall opinion of this drug or similar ones in public forum), reputation and trust (e.g. specific data or sources can have higher reputation among scientists).
6. Integration issues

6.1. Introduction

The integration problem is considered an open topic still pending of solution. Software systems were usually designed to work in isolation. During the last years, new requirements and challenges have appeared due to the evolution and improvement of the communication networks. At present, systems frequently need to exchange heterogeneous data and collaborate with other applications. But integration is a complex problem depending on many factors such as system architectures, operating systems, type of the components and information to be integrated, coupling and use of the systems, performance requirements, data heterogeneity and semantics, user interfaces, middlewares, and availability of resources (Ziegler and Dittrich, 2004). In the following sections, we analyze the state-of-the-art on integration issues from two different points of view, namely data (Section 6.2) and applications (Section 6.3)\(^\text{10}\).

6.2. Data Integration

Nowadays, it is difficult to imagine a modern organization, company or institution storing all their data in just only one system. Usually, they have the information distributed among several physical devices—i.e. computers, hard disks, databases, CD/DVD, etc. Efficient integration of such information becomes crucial in order to perform analysis, experiments or decision-making tasks. The underlying idea of data integration is simple: an organization has interrelated information in different places and it wants to retrieve all that information in a uniform way just making a unique query. But such simplicity is far from reality. Data integration has to consider from technical issues—e.g. computers features, database managements systems or communications—to problems related with the representation of information—e.g. information coding, representation models or data heterogeneity. Data integration is a still evolving complex discipline which needs much more research. Today, no general solution to integrate all kind of systems exists.

Traditionally, two major approaches to integrate data are considered: Centralized versus Distributed/Federated approaches. In the literature, such approaches are also known as Data Translation and Query Translation respectively. The main difference between them essentially lies in the physical place where data is stored and the methods and technologies used to retrieve such data. In the following sections, we will analyze in depth the features of each approach, detailing strengths and weaknesses and illustrating their use with relevant examples. Apart from these approaches, we can consider another data integration approach called Information Linkage. It is closely related to web environment where information is integrated using static hyperlinks (Anguita et al., 2010). But this approach does not constitute real integration. It is just collections of links regarding web pages about the same topic. Examples of this type of integration are MEDLINE (Lindberg, 1990), PDB (Berman et al., 2000) or Prosite (Hulo et al., 2006).

\(^{10}\) It is noted that integration in Dicode is addressed at both the architectural level (Section 6 of this deliverable) and ontological level (deliverable D5.2).
6.2.1. Centralized approaches

The most representative example of centralized approaches is the data warehouse. A data warehouse is often a database management system which gathers data coming from several databases. Data are imported into the data warehouse using a common format. To carry out this task, data needs to be transformed from the original format to the new one. Such transformation process is performed by an entity/program called ETL (Extract, Transform and Load). Those processes are usually executed in background mode. A typical data warehouse architecture can be seen in Figure 5.

![Figure 5: Simplified data warehouse architecture.](image)

The main characteristic of data warehouses is that data are physically stored in a common database. Therefore, results from queries launched against the system are quickly answered. Such performance and efficiency constitutes the major benefit of centralized approach. The major drawback lies in the potential size of the data warehouse especially when many data sources are integrated. Another non-trivial problem is how to keep continuously updated the central repository. Due to their own nature, data warehouses do not provide always updated and accurate views of the data sources. Updates are executed periodically but not every time something changes in data sources. Therefore, there exist periods where data warehouses offer an outdated view. In some scenarios such kind of situation might be acceptable but in others, it constitutes an unaffordable solution. The centralized approach is particularly suitable for systems which do not change frequently. Many examples of data warehouses systems in the biomedical area can be found in the literature (Etzold and Argos, 1993) (Maglott et al., 2007) (Shah et al., 2005) (Lee et al., 2005) (Xinxing and Wang, 2009).

6.2.2. Distributed / Federated approaches

Compared to centralized approaches, the main characteristic of distributed systems is that data remain physically stored in their original databases. Each time some information is required, data are directly retrieve from the sources. This task is usually carried out by a middleware or mediator layer which deals with syntactic and semantic heterogeneity of the data and the data sources. Whenever a query is launched, the mediator decomposes it and sends the appropriate sub-queries to all databases which are affected by the original query. Such decomposition is performed according to an existing global schema and the relations established between the global schema and the databases. Such relations are called mappings. Generation of sub-queries requires some transformations of the original query to be understood by the underlying databases. In some approaches, there are a kind of adapters, called wrappers, which facilitate the communication between the mediator and the databases.
Finally, the mediator collects all results, unifies them and returns an integrated result. Figure 6 shows a simplified schema of a distributed system.

![Figure 6: Simplified architecture of a distributed system.](image)

This kind of systems can be highly complex. Dependency from communication networks and access times are the main drawback of this approach. On the other hand, coherence, consistency and accuracy of data are warranted. The responsibility for maintaining the data and keep them updated is divided among all databases. Information retrieved is always the most current.

According to (Sujansky, 2001) and (Anguita et al., 2010), query translation approaches can be classified into four categories:

- **Pure Mediation**. The key components of these systems are the *wrappers*. The *wrappers* are software applications which act as mediator between databases and query systems. They hide the syntactic issues to the query systems, providing a homogeneous interface. Each database to be integrated into the global system needs one different *wrapper* and such *wrappers* are completely different among them. Examples of pure mediation systems are TSIMMIS (Molina et al., 1997), DISCO (Tomasic et al., 1998), DIOM (Liu et al., 1997), HERMES (Adali et al.,1996) or BioDataServer (Freier et al., 2002).

- **Global as View (GAV)**. These systems define a global conceptual schema from the particular schemas of the databases integrated—i.e. the global schema is the combination of all database schemas. Once the global schema is created, a mapping has to be established between the global schema and each database. The main benefit of this approach is that the global schema describes very well the underlying databases and a common vocabulary is shared by all databases. On the other hand, any change in the structure of the databases or adding new databases to the system force to reconsider the whole global schema. Examples of systems following this approach are SIMS (Arens et al., 1999), Ariadne (Knoblock et al., 2001) or TAMBIS (Stevens et al., 2000).

- **Local as View (LAV)**. Systems following the LAV approach define multiple conceptual schemas, one per database integrated. They have also a global schema but, each conceptual schema (mapping) is expressed according to the global schema. In this case, a common vocabulary is not shared by all databases. Accuracy and reliability of systems depends on how good the global schema is (in terms of representing the contents of the databases). This approach enables a high extensibility
and modularity but queries require more complex processing. An example of system with LAV is OBSERVER (Mena et al., 1996). Figure 7 shows a comparison between GAV and LAV approaches.

Figure 7: Comparison between GAV and LAV.

- **Hybrid approaches.** This approach tries to combine and take advantage of the benefits described in the previous approaches. Multiple particular schemas are created according to a global schema but using a common vocabulary shared by all. Examples of hybrid systems are SEMEDA (Köhler et al., 2003) or ONTOFUSION (Pérez-Rey et al., 2006).

### 6.3. Applications Integration

There exist thousands of applications and services already developed around the world. This number grows exponentially every day. Applications and programs need each others to carry out their work, exchanging all kind of data. Software integration architectures enable reusing existing applications and services to work together with new developments. Reusing applications have many benefits, from reducing cost to shorten the development time.

Software architectures must clearly define the organization and relationships among components and the environment. Additionally, such architecture should remain independent from tools or programming languages. Such independency enables that systems are implemented by several groups across the world.

There are several architectural styles for software development according to different aspects such as communication, domain, relationship, structure, data, data flows or object oriented. For Dicode project, we have evaluated the most relevant software architectures. Due to the nature of the project, we have focused our analysis in distributed architectures.

- **Remote Procedure Call (RPC):** is one of the foundations of distributed computing, rooted in the popular programming procedures (Birrell et al., 1984). These procedures are a mechanism of control and data within a program running on a machine. However in the case of RPC these procedures are hosted on remote machines so that control and data transfer is performed through a network communications.

The operation in the case of the RPC is as follows: when the remote procedure call starts, the local environment of the procedure is suspended. The parameters are sent through the network to the environment of the procedure being invoked and this runs on a remote machine. After completing the execution, results are sent back to the local environment, which resumes its own execution. It is noteworthy that although
the original process is suspended, other local procedures may still be running by the machine to run efficiently.

The main purpose of RPC was to achieve distributed computing by simple procedure calls to bring the technology to developers who are not experts in the field.

- **Remote Method Invocation (RMI):** is a mechanism for invoking remote procedures offered by Java. It is characterized by ease of use for expert developers in Java (Waldo, 1988). The way to create such procedures is to define an object that can be used remotely. An RMI remote interface is defined, extending the `java.rmi.Remote` interface. On the side, a server is waiting to be invoked. In this way, programmers can create items that invoke the object through a reference.

The RMI protocol, while simple, also carries an important handicap, since this technology limits the programming language to Java, excluding applications written in other languages.

- **Common Object Request Broker Architecture (CORBA):** is a standard that facilitates the development of distributed system, enabling integration of applications written in different programming languages and platforms by invoking remote methods following the object-oriented paradigm (Vinoski, 1997)(Vinoski, 2002).

The Object Management Group (OMG) defined APIs, communication protocol and other mechanisms to ensure interoperability. For exchanging data, CORBA objects define contracts with an interface definition language (IDL) that contain the information necessary to generate client and server code that reflects the features offered by the CORBA object.

- **Event-based Architectures:** Event based architectures rely on implicit invocation. This means that components do not invoke directly the procedures, but they are waiting until one component launches one or more events. Such events are captured by the receiver modules that previously have registered themselves in the event producer. Events producers do not know how is interested in events. Such independency allows updating components in a transparent way and reducing coupling among components. Systems based on events facilitate extensibility and reusability of components. The mayor handicap of this approach is that components waiting for events cannot know the current state of the system. This kind of technology has been mostly applied in local networks environments, although they have been used over the Internet.

- **Service-Oriented Architectures (SOA):** rely on the use of services which provide support to the institutions involved in the project. These services are loosely coupled with each other but they are highly interoperable thanks to a simple interface, which makes them completely independent of the platform where the services are developed (Erl, 2004).

Services are deployed through the Internet using web services. These services are developed according to requirements specified in a contract which must be met by both, clients and the service provider. That contract states how to make the calls, information to exchanges, etc. Developers can implement applications in any platform, language or tool that support these services without knowing anything about how those services were created.

The use of services enables high reusability on the client side; nevertheless it is barely flexible in case of contract modifications, because the smallest change makes the client programs useless.
Related to SOA, few years ago appeared the Enterprise Service Bus (ESB) approach (Schmidt et al., 2005). Basically, it defines an infrastructure to implement SOA-based systems using a middleware layer to enable message exchanges between resources. To facilitate the communications, messages can be transformed by the middleware layer from source format to target format. Additionally, such middleware layer provides advanced features like load balancing or failover.

- **Resource-based Architectures (REST):** also known as REST (REpresentational State Transfer) can be seen as an evolution of SOA. It tries to solve problems in which SOA seems to be too rigid in terms of changes and representations. For instance, in the case of REST, the contract is just for the inputs.

  The main element of REST is not based on services nor calls, but resources that are available via URI (Uniform Resource Identifier). It relies on HTTP protocol to access to the resources by using a set of operations such as POST, GET, PUT and DELETE.

  Another advantage of REST compared to SOA is that resources can be represented in multiple formats such as XML, HTML, text, pdf, pictures, etc., whereas in SOA the exchange is completely based on XML files. Despite these advantages, REST is not completely standardised and its background and specifications are not as mature as SOA.

In the next sections, SOA and REST are analysed in more detail because they constitute the current state-of-the-art in integration technologies. It is pointed out the pros and cons, standards and languages used and other relevant issues. This information will help to decide which architecture is the most suitable for fulfilling the needs and requirements established of the project.

### 6.3.1. Serviced-Oriented Architecture

Software systems require a proper integration between their several modules and components. To ensure a successful integration, there are two factors that must be taken into account: coupling and adaptation to standards. These factors may hamper the tasks of designing and implementing software systems, converting them into products that do not scale and hence will not be used.

The coupling of a system is given by the degree of interdependency among modules and programs. It is desirable that this interdependency remains as little as possible because a loose coupling between components facilitates the modification of any of the modules without affecting the rest of the parties. If the modules are more dependent on each other, it is more complicated to integrate a module into another system without having to interact with all modules.

Adaptation to standards relies on the correct design and documentation of the system. Well-planned and designed systems have interfaces for integration between its modules. By using standards, the need to develop specific software to perform this integration is minimized.

Service-oriented architecture (SOA) is based on well-defined standards. It is focused on the low coupling between modules of the system. SOA is not a tool, technology or product, but a concept, a set of rules and principles to design software, regardless of the technology used in its development. SOA relies on the creation of some interfaces that abstracts away its underlying complexity. By using such interfaces, clients and providers may establish communications, just knowing the inputs and outputs of the services.

SOA is usually implemented using web services. A web service is a set of standards and protocols which allow the information exchange between different applications. They
provide low coupling and adaptation to the standards demanded by SOA. Due to its nature, web services are especially appropriated to implement SOA standards. However, we could apply SOA without web services. Applications using SOA can be found in distributed environments. They communicate with each other through the interfaces to obtain information or execute a particular workflow. These applications are platform-independent, i.e. they can be developed with different tools, languages and platforms.

Standards followed by web services are SOAP, XML, WSDL and UUID. Table 5 summarizes the standards used in SOA.

<table>
<thead>
<tr>
<th>Task / Functionality</th>
<th>Language / Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>XML</td>
</tr>
<tr>
<td>Format</td>
<td>SOAP</td>
</tr>
<tr>
<td>Transport</td>
<td>Technologies Web (http, smtp, etc)</td>
</tr>
<tr>
<td>Interfaces</td>
<td>WSDL</td>
</tr>
<tr>
<td>Find / Search</td>
<td>UDDI</td>
</tr>
</tbody>
</table>

**Table 5**: Standards and languages used in SOA.

![Diagram](image)

**Figure 8**: SOA’s lifecycle

Figure 8 shows the life cycle of a service-oriented architecture and the standards involved in each stage of the process.

Next, we detail the characteristics of the standards and protocols depicted in the diagram.
**XML**

XML stands for eXtensible Markup Language, founded in 1998 with the goal of becoming a structured language, extensible and validated with the aim of enabling the transmission of structured information. XML is not functional. It was created to structure, store and transport information. XML is a meta-language that is used to define other languages by means of marks. There are two basic mechanisms to validate a XML file: DTD (Document Type Definition) and XML Schemas. XML documents contain a hierarchical structure and strict rules to compose documents. XML is both human and machine readable.

![XML document structure](image)

**Figure 9:** XML document structure

The XML document structure is very rigid (Figure 9). It consists of a prologue and a root element. It should be noted that XML documents can have comments that have been contained between the symbols `<!--` defining the beginning and `-->` that marks the end of the comment. It can be placed at any time except within the definition of an element or attribute.

In the prologue to the document appear processing instructions and information about the document format such as the version (1.0 or 1.1) and its encoding (UTF-8, UTF-16, iso-8859, etc.).

Labels composing the XML document are expressed by pairs, with an initial label and a final label—e.g. `<Name> ... </Name>`. Labels can contain nested elements, thus forming the document hierarchy. Names containing special characters such as underscore (_), dash (-), colon (:), colon (-) or numbers are allowed, but names starting with digits are forbidden.

Labels may have attributes, defined within the initial label, as a pair attribute-value—e.g. `<Element language="spanish"> ... </Element>`. Attributes support prefixes that are placed before the description and followed by `:`—e.g. `<Element xml:lang="sp"> ... </Element>`. Note that in any case, the attribute value must be enclosed in double quotes.

**SOAP**

While SOAP stands for Simple Object Access Protocol, commonly this acronym also is linked to Service-Oriented Architecture Protocol, but in the current document we will always refer to the first one. SOAP comes from a previous protocol, XML_RPC, which allows remote procedure calls (RPC). Clients or servers were able to make requests to a web server via
HTTP. That protocol resulted to SOAP due to the interest of multinationals such as IBM or Microsoft among others.

SOAP protocol establishes a standard format for message exchange throughout an XML document; it was created by W3C and is formally defined as follows: “Light protocol for data interchange within a distributed and decentralized architecture”. In addition, SOAP messages can attach information to the XML messages. SOAP does not take care about reliability, integration, transaction or security and safety.

Figure 10: SOAP message structure

SOAP is not associated with any language or any transport protocol. Therefore, developers might implement APIs supporting SOAP in any language. Messages can be sent using any transport protocol capable of transmitting text. These characteristics make it highly portable and reusable.

SOAP messages have a very simple structure, divided into two sections: header and body (Figure 10). The entire message is wrapped in a SOAP envelope section that covers the header and body. Such section is the root element of SOAP message and identifies the XML document as a SOAP message. The header is the first element found within the SOAP envelope and contains meta-information about the message. It is optional, but if present, it must appear immediately after the root element “envelope”. Header consists of several elements called “header blocks” that allow adding new capabilities to SOAP messages. The body element contains the information to be exchanged between the client and the service. It may contain the request of a service or the results.

WSDL

WSDL stands for Web Services Description Language. It is an XML format to describe Web Services. The document lists the requirements of the protocol and message formats required to interact with web services. It is usually used in combination with SOAP and XML Schema. Definitions are given in XML and the client can use SOAP to invoke functions listed in the WSDL.

WSDL structure is clearly differentiated and its documentation can be found in the recommendations by the W3C, both versions 1.1 and 2.0. The document is structured as can be seen in Figure 11.
Figure 11: WSDL document structure

The document is divided into two main sections which are: i) the abstract section, and ii) the concrete section. The abstract section defines data types and structures, messages and groups of operations. In the concrete section, what the service does is stated in detail. It specifies ports, transport protocols, service location and encoding for messages.

The description is based on the model 1.1 as version 2.0 is not yet supported on some servers. In any case, versions are quite similar. Some labels have been renamed and some elements—such as the “message” block”—have disappeared. The 2.0 model attempts to address some existing gaps like the quality of service, security service or compositions, that were missing in version 1.1.

Looking deeper at the different sections of the WSDL document:

- **Data Types**: in this section are defined the types of data used by the service.
- **Messages**: this section defines the elements of the messages providing abstract information for communication between the client and the server. The messages contain one or more elements “Part” to describe the message contents, according to the data types previously defined.
- **Port Types**: compile a set of operations under a common name. WSDL 2.0 has replaced this section with the *interface* section. Actually, this section has no functionality itself. It is just a collection of operations.
- **Operations**: this section contains abstract descriptions of operations that are supported in the Web Service. There are four types of operations:
  - *Request-response* corresponds to a communication in which the client makes a request and the server sends the reply.
  - *One-way*, in this case the client sends a message to the server, but is awaiting any kind of response.
o *Solicit-response* is similar to the request-response, but in this case, the server is requesting a client request and the latter is the one that sends the response.

o *Notification*, is the opposite operation to *one-way* operation. The server sends a communication to the client without any previous request.

- **Bindings**: the binding elements contain the definitions of specific transport protocols used to access the service. They define how abstract messages are transformed to specific data types used in the access point.
- **Services**: this section consists of a collection of related ports. Optionally, it can be added a textual description tag to document the service.
- **Ports**: it establishes the access point to the physical service in a binding and defines a single destination point—i.e. it associates the binding information to the URI where the service implementation can be accessed.

**UDDI**

UDDI (Universal Description, Discovery and Integration) is a protocol to interact with a server that provides mainly operations to search for web services. It defines a directory and a data model for storing information and business services. Registering a business in UDDI consists of three sections:

- **White Pages**: provide identification and contact address of the company or organization that publishes the web service.
- **Yellow Pages**: describe the web services offered, classified according to different categories or taxonomies.
- **Green Pages**: provides technical information and specifications regarding the web services.

Interfaces defined using UDDI can be used by both providers and consumers of web services. Such interfaces are described in WSDL and they can be accessed via SOAP messages using a transport protocol such as HTTP. Regarding the publication of web services, UDDI operation begins when a developer builds a service that wants to share with other people. Developer has to register and classify the web service in a UDDI registry. Once the web service is registered, it can be accessed by any user through its WSDL definition.

6.3.2. **Examples of technologies to implement SOA**

There exist several platforms—both public and private—for implementing SOA based on web services. For instance, Microsoft offers the .NET platform for the development of proprietary software. The J2EE platform from Oracle is an open source solution for the development of these types of services. Through the .NET platform, we are offered the C# language for development, while the J2EE platform requires the use of the Java language. We can create a web service in Java relying in the SOAP implementation of Apache, or in C# with the .NET Framework and the Visual Studio development environment.

The Apache Software Foundation includes a project (Apache SOAP projects) that offers enough resources for implementing any service based on SOAP through Axis. The new version of Axis—Axis2—provides support for both Java and C (the previous version also supported C++). Axis is the most extended technology for the development of web services, and it is worth pointing out that it works with the Jakarta Tomcat application server (also from Apache). Apache offers in addition several projects related to web services, all of which can be checked in the following URL: http://www.apache.org/soap.
Additionally, in the case of Java, we can use several available tools: the Java API for XML web services, JAX-WS (with GlassFish), Zope—a web application server developed with Python—, or JBoss—belonging to J2EE and Red Hat inc.

6.3.3. Examples of SOA technologies in the field of Bioinformatics

There exist a great number of projects belonging to the area of bioinformatics that have been developed under the service oriented architecture. It must be noted that, in order to develop a project under SOA, there must exist a web service provider and one or more clients that develop applications for using such services. Below, some examples of web services are given.

**National Center for Biotechnology Information (NCBI)**

Without any kind of doubt, the most important example to take into account is the NCBI (National Center for Biotechnology Information) portal, available at www.ncbi.nlm.nih.gov. It provides access to some of the most important databases in biomedicine. The NCBI has developed the so called Entrez Utilities (EUtils), a set of web services for accessing the resources stored at their databases from any environment. It even allows developers to create new tools that use these resources. This way, a great number of projects have been carried out benefiting of the SOA architecture offered by the NCBI.

The NCBI website provides developers with the WSDL that describes the various services offered by the portal for client development. WSDL document can be downloaded from http://www.ncbi.nlm.nih.gov/entrez/eutils/soap/v2.0/eutils.wsdl. It allows accessing to all features of EUtils unless Efetch. Efetch exists individually for some specific bases and likewise their respective WSDL can be downloaded for each of them.

Also, NCBI offers program clients to programmers in different precompiled code for ease of use in languages like C#, Visual Basic and Java. This code comes with tutorials and documentation sufficient to be able to start easily any project. Notably, as well as SOAP based web services, NCBI also provides access to data via REST offered on its portal.

**European Bioinformatics Institute (EBI)**

The European Bioinformatics Institute (EBI) was the pioneer in bioinformatics research. The principal center is located in Cambridge (UK). The EBI hosts many databases with information on genomics, proteomics, nucleic acids, proteins and macromolecular structures. All the resources are free and unfettered and may be downloaded from the EBI website by anyone. EBI is committed to providing reliable and updated data that have gone through rigorous quality processes and facilitate the work of developers sharing resources and structuring the databases through links between them.

EBI, like NCBI, also offers developers a range of web services to manage their databases. They can be found in http://www.ebi.ac.uk/Tools/webservices. In that link we can find all WSDL enough to generate customers and create our own tools that make use of the resources offered in EBI website. The website has available different tutorials, documentation and examples that simplify the task of software development, even for not experienced users.

**BioMart**

BioMart is a project developed by the Ontario Institute for Cancer Research (OICR) and the European Bioinformatics Institute (EBI) that integrates seamlessly different types of data marts through a web interface. It consist of several relational databases to which they can query through several APIs, including one written in Perl and one in Java. Through Perl API, we can use MartView (web interface), MartService (web services interface) or
MartURLAccess who access MartView through URL. In the case of the Java API, we can use MartExplorer, a standalone GUI tool or MartShell, a command-line tool.

Apart from these implementations offered by BioMart, the website provides a range of web services defined in a WSDL (http://www.biomart.org/biomart/martwsdl) for those who want to generate their own client programs and access to services offered by BioMart from a proprietary tool.

**Other Examples**

CaBIG (www.cabig.cni.nih.gov) is an initiative, which collaborates with the NCI Center for Bioinformatics, providing a robust web service oriented architecture focused on assisting cancer researchers.

We can find numerous projects that provide services for the integration of web services in a project, such as myGrid (www.mygrid.org), which also allows using workflows and distributed query processing. DAS (www.biodas.org) is a system for client-server distributed annotation in which any client can integrate information from different servers. A similar example is BioMOBY (www.biomoby.org), a web service repository that allows building SOA projects using their services.

### 6.3.4. Resource Oriented Architecture: REST

REST is an acronym that stands for **REpresentational State Transfer** (Fielding, 2000). REST is an *architecture style* based in the resources that are provided by the World Wide Web. Roy Fielding developed these concepts in his PhD thesis: “Architectural Styles and the Design of Network-based Software Architectures”. It was defended in the University of California – Irvine in 2000.

REST contributes with a set of constraints for designing network architectures. Thus, REST is *not an exhaustive standard* applied to network applications but an architectural style that is supported by existing standards such as: URI, HTTP and HTML. Indeed, Roy Fielding was coauthor of the HTTP specification while working with the W3C in the definition of standards for the World Wide Web, which have been quoted before. The idea of Fielding was to represent the behavior of a web application that could be taken as a model because of its design. In this context, the web applications that caught the interest of Fielding were those ones that fit the model of *distributed hypermedia system*: any kind of resources (text, image, video, other files...) linked each other through hyperlinks and placed or distributed in different servers over the network.

In the following, we detail the standards and protocols used in REST.

**HTTP**

HTTP is the protocol used in the World Wide Web for communications (Fielding, 1999). It is a network protocol placed in the Application layer of the OSI model and designed for distributed hypermedia systems. It uses the *URI* (Uniform Resource Identifier) standard to reference a resource as a location or as a name. Messages are formatted following the *MIME* specifications. The HTTP works as a request-response protocol; the client submits a request through a HTTP message (containing the URI to reference the resource and a method to be performed in that resource) and the server sends back the response message (containing a status code, a reason phrase and the message itself). Different methods allow the client to perform different operations on the remote resources such as: get, put, delete, connect, etc. Every resource can define what methods are allowed to be performed on it. At the moment, the last version of the protocol is known as HTTP/1.1.
URI

URI stands for Uniform Resource Identifier (Lee, 2005). These resources could be names, locators or even both. To identify uniquely a resource could be used two subsets of URI:

- **URN (Uniform Resource Name)**: used to identify uniquely a resource.
- **URL (Uniform Resource Locator)**: used to identify the location of a resource in a network.

The typical syntax would be: `<scheme>://<authority><path>?<query>`.

MIME

MIME stands for Multipurpose Internet Mail Extensions (Freed, N. and Borenstein, 1996). MIME extended the RFC 822 standard. The main additions to that standard were to extend it to character sets different from US-ASCII and message bodies different from text. This standard also enables sending multi-ASCII message bodies. Initially, it was designed to describe the contents of e-mail, but after that it became broader and also covered the description of contents of HTTP messages. A MIME message contains the following header fields:

- **MIME version**: version number of MIME that the message is using.
- **Content Id**: used for multipart messages. It is a unique indicator for every part of the message.
- **Content type**: plain text, images, audio, video or application documents. It describes what is contained in the message.
- **Content disposition**: as a consequence of the multipurpose use of the MIME messages, the header fields had to describe the presentation style of the content of the message:
  - **Inline content disposition**: the message will be displayed as it is, without any intervention from the user.
  - **Attachment content disposition**: the message will not be displayed as one message; the attachment is not a part of the main message. The attachment will be displayed under user request.
- **Content transfer encoding**: indicates the method used to transform the binary data into text format.

XML

This standard has been presented in an earlier section and we refer to section 6.3.1 for a detailed description.

WADL

Web Application Description Language (WADL) is used to describe applications based in the current architecture of the World Wide Web, that is, oriented to REST architectures (Hadley, 2006). It could represent the version of WSDL (Web Services Description Language) designed for REST, however the WSDL 2.0 (Chinnici et al., 2007) (Mandel, 2008) version is suitable for describing REST web services, thus WADL can be seen as a competitor for WSDL.
6.3.5. Examples of REST applications

TogoWS

Some people associated to the University of Tokyo and other institutions in Japan developed TogoWS (Katayama et al., 2010). This service aims to integrate different interfaces into a common interface with advanced features. They provide both SOAP and REST interfaces to access the most renowned bioinformatics services in the web provided by institutions such as the National Center for Biotechnology Information (NCBI) in the USA, European Bioinformatics Institute (EBI), DNA databank of Japan (DDBJ), Protein databank of Japan (PDBJ), Kyoto Encyclopedia of genes and genomes (KEGG) and Computational Biology Research Center in Japan. The architecture of TogoWS is shown in Figure 12.

The diversity of interfaces and data types in the different Bioinformatics web services makes it difficult to use them together to obtain the desired results. Togo WS enables users to access several Bioinformatics web services deployed by the institutions previously mentioned.

Thus, Togo WS provides a common interface to access in black box mode the abovementioned web services. Users communicate with Togo WS through a REST or SOAP interface.

Biocatalogue

BioCatalogue is a repository of biological web services deployed using an open platform that every user can access freely (Bhagat et al., 2010). It is a project led by the University of Manchester and the EMBL-EBI. The project is strongly oriented to the Web 2.0 principles: collaborative environment, REST API, XML to make easier the information exchange, etc.

This project aims to homogenize the way to search and to use biological web services on the Internet. Biocatalogue provides a standardized way to publish web services, so that users can use them easily. Currently, Biocatalogue provides access to 1741 services, 140 services providers and 469 members.
caBIO (National Cancer Institute)

caBIO stands for Cancer Bioinformatics Infrastructure Objects (Covitz et al., 2003). It is a project supported by the National Cancer Institute of the U. S. Institutes of Health located in Maryland, USA. caBIO enables users to access its services through a REST API. Users make queries via HTTP and get results in XML/HTML under request. The architecture of caBIO is shown in Figure 13.

The functionality of caBIO is described in its webpage (http://cabio.nci.nih.gov/): “Conducting biomedical research requires access to experimental data, as well as associated molecular annotations. Annotations providing detailed information on the molecular origin, biological process, and genetic alterations can provide important insight on experimental outcomes. caBIO (cancer Bioinformatics Infrastructure Objects) is a robust resource for accessing molecular annotations from curated data sources in an integrated view in support of knowledge discovery. The entities that concern the Central Dogma of Molecular Biology are the core of the model.”

![Figure 13: caBIO architecture](image)

6.3.6. Comparison: SOA vs. REST

REST appeared as an evolution of SOA to solve some problems of the latter in certain projects. REST implements web services ignoring the SOAP protocol. However, there are certain characteristics that can benefit one from each other, depending on the type of project to develop. The main difference between REST and SOAP is given on the basis of both architectures; REST gives more importance to information while SOAP is opting for message exchange. Below, a comparison between web services based on SOAP and REST is presented (Pautasso et al., 2008).

- **Technology**: while SOAP offers a multitude of operations that operate on limited resources, REST instead offers a few operations that can interact with a multitude of resources. These operations in the case of SOAP are given by a flow of events. In the case of REST interact directly with users via forms. REST has a consistent mechanism for naming resources through URI whereas SOAP does not have one.
- **Protocol**: SOAP XML documents are strongly typed based on XML Schema, while REST uses a self-describing XML. Although both technologies use HTTP as transport protocol, SOAP might use other. HTTP is also used by REST as application protocol. Finally, REST works synchronously but SOAP can work both synchronous and asynchronously.
• **Service Description**: SOAP is based on contracts using the WSDL standard and REST establishes a set of documents user-oriented defining the directions of requests and responses. WSDL documents (from SOAP) are more difficult to understand by humans than REST definitions. Besides, WSDL allows building automatically web services clients from the descriptions. Since November 2006, REST includes the WADL standard to emulate SOAP mechanisms.

• **State management**: Both approaches enable state management but use different methods. Since REST servers are stateless—i.e. servers do not remember previous queries or invocations—each request must contain all the information needed to answer the petition. State transitions can be simulated using cookies in the client side and incorporating extra data and links to resources through the URIs or in the message payload. On the other hand, SOAP can maintain the current state of the services on the servers using sessions, although SOAP session headers are not standard. Additionally, SOAP may communicate such information using message payload or WS-Addressing specification.

• **Security**: SOAP provides security mainly through WS-Security protocol. This protocol defines how to provide integrity and confidentiality on exchanged messages and how to attach security tokens, such as SAML, Kerberos and X.509 certificates. It is associated with other specifications such as WS-Policy, WS-Trust or WS-SecureConversation, WS-Federation, WS-Privacy and WS-Test. Security in REST is based on the mature HTTPS protocol to provide point-to-point SSL secure communications. This protocol can also be used in SOAP as complement of the WS-Security standards. REST does not have defined open standards comparable to WS-Security for distributed transactions. Therefore, the use of proprietary security implementations is needed when HTTPS is not enough to provide digital signatures or detailed authentication and authorization.

• **Design methodology**: to design applications is important to consider that REST is focused in resources and on the other hand, SOA is focused on message exchange. In case of REST, we have to consider what information and resources will be available as services, while in SOA, we have to identify the operations appearing in the WSDL document. In REST, it is necessary to define different URLs to address the web services, but in SOA, we must define a data model for the content of messages. Finally, in SOA we need to implement, register and deploy the web services, while in REST it is only needed to implement and deploy the web service without any registration.

### 6.4. Implications for Dicode

Integration is still an open and complex problem affecting all scientific disciplines. During the last years, eHealth disciplines—i.e. Medical Informatics, Bioinformatics, Biomedical Informatics —have been especially active and productive in the development of a plethora of resources. Integrating such resources by using uniform access methods has become a crucial issue to improve research results. Integration does not only involve data integration but also the integration of systems, applications and services.

There are two major approaches for data integration according to the physical location of stored data and how these data are retrieved:

- **Centralized approaches** - represented mainly by data warehouses. Data coming from different databases are retrieved from their original sources, transformed into a common schema and stored into a unique and huge database. The major benefit of this approach is that data can be retrieved quickly but maintaining up-to-date data is a complex task.
- Distributed approaches - where data remain in their original location and are retrieved when needed. A mediator/middleware software is used to deal with the different sources and to unify the results. Access to the information is more complex but consistency, coherence and currency of data are maintained at source.

Regarding the integration of applications and services, many approaches have appeared during the last decades. Some of them depend very much on the implementation language—e.g. RPC or RMI. The most recent approaches define a generic public interface to facilitate the interoperability—e.g. CORBA, SOA or REST. Current trends are committed to the use of service-oriented architectures such as SOA and REST.

Dicode is a highly interdisciplinary project dealing with very heterogeneous data. According to the definition of the project, for data integration, we envisage using a distributed approach to homogenize data access. Such access would be based on a mediation layer supported by the Dicode Service Ontology (DSO). Regarding applications and services integration, we envisage the use of a service-oriented approach also supported by the DSO. SOA and REST approach seems the most appropriate for Dicode. There are a lot of resources already developed using these technologies. We envisage reusing as much resources as possible, trying to avoid reinventing the wheel. In Dicode, we also plan to investigate the use of workflow environments such as Taverna, Kepler or Triana and will evaluate the suitability of the ESB open source frameworks with respect to Dicode’s purposes and objectives. SOA and REST technologies are particularly appropriated to be integrated into such systems.
7. Summary and Conclusions

The Dicode project aims at further advancing the current progress in knowledge modelling and processing by developing innovative commercial and community services that enable the conceptualisation and production of richer digital content, which can be individually or collaboratively selected and manipulated. To achieve this goal, Dicode will exploit and significantly advance the state-of-the-art in the directions that have been thoroughly elaborated in this deliverable (see Sections 3-5), namely: scalable high-performance data mining, data mining towards sense-making of real-world multi-faceted data, and collaboration and decision making support. As mentioned in Sections 3-5, several techniques and technologies for processing data into value added information that can be turned into knowledge are currently available and used in various situations. As yet, most of these lack either in terms of scalability or in terms of completeness. Moreover, the appropriate integration of these technologies (see Section 6) will enable the desired synergy of human and machine intelligence, to which the project gives equal importance.

The project’s progress and achievements will be validated through three use cases, which have been carefully selected to address clearly established and contemporary problems, while representing alternative collaborative decision making paradigms and dealing with various types of large scale and real time data residing in heterogeneous sources. It is expected that the detailed elaboration of the Dicode approach, which is to be reported in deliverable D2.2 (entitled “The Dicode approach: User requirements, conceptual integrative architecture, agile methodology and functional specifications”), will further clarify the project’s broad research directions. In any case, the foreseen advancements with respect to the state-of-the-art issues discussed in this deliverable are summarized in Table 6.

From a more abstract point of view, we distinguish three major target outcomes, as far as the project’s research and applied work is concerned. The first one concerns the capturing of tractable information. Focusing on diverse data sources, Dicode will elaborate and extend the most prominent large data processing technologies to acquire, analyse and categorise extremely large, rapidly evolving and potentially conflicting and incomplete amounts of information. These services will enable users working in data-intensive and/or cognitively-complex settings to extract actionable meaning from structured and unstructured information and social interaction patterns, as well as to correlate data from diverse sources and formats while tracing their provenance and assessing their reliability through appropriate semantics.

The second one concerns the delivering of pertinent information. Dicode will improve the efficiency of the information lifecycle, starting from proactive diagnoses of information gaps and triggering goal-directed search, acquisition, aggregation and visualization of relevant resources. The foreseen workflow-driven integration of the project’s services (see Section 6) provides the means for such improvements, which will be evaluated through project-specific instruments and metrics (to be developed in WP6: Validation & Assessment). In addition, data management in Dicode is highly associated with large-scale reasoning resulting in effective ranking and interpretation. The foreseen Dicode environment will facilitate the navigation and manipulation of digital information by means of adaptive user-information interactions.
<table>
<thead>
<tr>
<th>Area</th>
<th>Topic</th>
<th>State-of-the-art</th>
<th>Advances foreseen in Dicode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalable high-performance data mining</td>
<td>High-performance data mining</td>
<td>Single, problem-specific solutions</td>
<td>Comprehensive data mining framework that supports high-level data mining tasks</td>
</tr>
<tr>
<td></td>
<td>Scalable data mining</td>
<td>High ratio of manual work</td>
<td>End-to-end support of data analysis; support of semantic management of models and processes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No support for complex, structured data objects</td>
<td>Extension of the MapReduce approach to deal with complex data types; integration of complementary frameworks</td>
</tr>
<tr>
<td>Data Mining on unstructured and multi-faceted data</td>
<td>Data Mining to find interesting links / objects</td>
<td>Restricted to single data types</td>
<td>Supporting heterogeneous views on the data; linking heterogeneous, multi-faceted data</td>
</tr>
<tr>
<td></td>
<td>Text Mining</td>
<td>Needs large labelled training corpora</td>
<td>Interactive generation of data from the web; drastic reduction of training time</td>
</tr>
<tr>
<td></td>
<td>Opinion Mining</td>
<td>High effort due to largely hand-crafted systems</td>
<td>Increased efficiency, robustness and accuracy due to new, semantically rich text mining approaches</td>
</tr>
<tr>
<td>Collaboration support</td>
<td>Collaboration support</td>
<td>Current tools are “information islands”</td>
<td>Increased interoperability and synergy with third party tools</td>
</tr>
<tr>
<td></td>
<td>Web 2.0 collaboration tools are rather passive media</td>
<td></td>
<td>Intelligent reasoning services to actively and meaningfully support collaboration</td>
</tr>
<tr>
<td></td>
<td>Web 2.0 collaboration tools cope poorly with voluminous and complex data</td>
<td></td>
<td>Advanced decision making support services; building on the synergy of human and machine reasoning</td>
</tr>
<tr>
<td>Decision Making support</td>
<td>Decision Making support</td>
<td>Problem- centric view</td>
<td>Emphasis on human-centric view</td>
</tr>
<tr>
<td></td>
<td>No thorough exploitation of underlying knowledge</td>
<td></td>
<td>Knowledge-based decision-making view; building on the synergy of human and machine reasoning</td>
</tr>
<tr>
<td></td>
<td>Little attention to dialoguing and argumentation</td>
<td></td>
<td>Argumentation-based reasoning mechanisms</td>
</tr>
</tbody>
</table>

**Table 6:** The advances that Dicode will bring about.

The third major target outcome concerns the overall *collaboration and decision making support*. The Dicode project will develop efficient and dependable services that augment problem
solving, sense making and decision making support for critical, information-bound domains in which our ability to share and exploit information is surpassed by the rate of its expansion in both size and complexity. The abovementioned augmentation will result out of the meaningful and efficient integration and orchestration of all Dicode services, which has also to pay attention on scalability, flexibility and performance issues.
8. References


Bentahar J., Moulin B. and Belanger M. (2010), A taxonomy of argumentation models used for knowledge representation, Artificial Intelligence Review, Volume 33 Issue 3


Hastie, T., Tibshirani, R., Friedman, J. (2002). The Elements of Statistical Learning, Springer Verlag, NY.


MacKenzie J (1981), 'The Dialectics of Logic', Logique et Analyse NS 24, 159-177


MediaWiki homepage, http://www.mediawiki.org/wiki/MediaWiki

Melnik, S., Gubarev, A., Long, J. J. et al. (2010), Dremel: Interactive Analysis of Web-Scale Datasets In 36th International Conference on Very Large Data Bases (September 2010)


Owen, S., Anil, R., Dunning, T. and Friedman, E. (2011); Mahout in Action; Manning.


Xiao, H. (2010), Towards Parallel and Distributed Computing in Large-Scale Data Mining: a Survey, Technical University of Munich


Yan, F., Xu, N. and Qi, Y. (2009), Parallel inference for latent Dirichlet allocation on graphics processing units, In: Advances in Neural Information Processing Systems 22, pp. 2134-2142

Yang, X., Parthasarathy, S., Sadayappan, P. (2010): Fast Mining Algorithms of Graph data on GPUs, KDD-LDMTA’10, July 25, Washington, DC, USA
Yong Ong, H. (2010), Accelerating Data-intensive Applications Using MapReduce, MSc Thesis, School of Informatics, The university of Edinburgh, 2010

