Presença: Aplicações Conscientes de Contexto

João Peixoto

October 22, 2009

Abstract

It's undeniable the wide spread of mobile devices, from cellphones to laptops, PDAs or handhelds. This devices have the capability of easing everyday's life tasks, organize schedules, share information, allow anywhere-anytime communication and so on. This detachment from fixed locations has however major implications in software design. How can we understand the global behaviour of multiple, concurrent, independent and self-capable devices?

Context-aware software aims to simplify the management of simple systems (which require user-intensive interactions) or complex systems that require intensive management. But what exactly is a context-aware system? Context-aware systems are no more than applications, either in a network or in a device, that are aware of their surrounding environment and through which they are capable of adapting their operations dynamically and automatically in order to ease our interaction [7, 10, 1, 2].

The aim of this project is to provide a framework capable of gathering context from different sources, understand that levels of confidence are associated with such context and infer, based on dynamically defined rules, the best way to communicate with an user.

1 Introduction

Pervasive computing consists of countless "pervasive" devices equipped with limited resources and computing power that in conjunction support end-to-end applications which far exceed their own capabilities. The complexity of an application is directly proportional to the difficulty to manage it, therefore, the growth of pervasive computing is pushing software engineers towards the threshold of incapability to manage such systems.

Context-aware software aims to simplify the management of simple systems (which require user-intensive interactions) or complex systems that require intensive management. But what exactly is a context-aware system? Context-aware systems are no more than applications, either in a network or in a device, that are aware of their surrounding environment and through which they are capable of adapting their operations dynamically and automatically in order to ease our interaction [7, 10, 1, 2].

The aim of this project is to provide a framework capable of gathering context from different sources, understand that levels of confidence are associated with such context and infer, based on dynamically defined rules, the best way to communicate with an user.

We will present a brief analysis over existing techniques to represent and reason with context, as well as social groups and their information richness. We will refer to data merging too. We then present the project's architecture, focusing too on the information architecture behind the system. After some technology-related considerations we present the evaluation for this project and the final considerations.

2 Related Work

In this section we will analyse context-aware applications at several different levels. We will begin by exploring the concept *context* itself (see section 2.1), context representation techniques (see section 2.1) and different approaches to context reasoning (see section 2.1). Social groups (see section 2.2) will be covered with an analysis over real-world social group services. We will finalise with a perspective over extraction, transformation and loading (ETL) of information (see section 2.3).

2.1 Context-Awareness

Context is all about the whole situation relevant to an application and its set of users. We cannot enumerate which aspects of all situations are important, as this will change from situation to situation. Dey[6] produced a very accurate definition of context that takes into account these concerns.

Definition of Context

> "*Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.*"

It becomes much easier to assess context information in a specific situation/interaction using this definition, and through that assessment designing and prototyping of applications are largely empowered, but still, we must be able to identify what is important for each interaction, rather than using all information available, even if it is specific to an entity.

- **Situation 1**: Alice asks Bob to close the door. Bob will (correctly) assume that the door to be closed is the one from the room where they both stand in.

Lets analyse situation 1 using Dey's definition. The actors of this interaction are (roughly) Alice, Bob, the door and the room itself. These entities have properties

such as age, mood, color, clothing and so on. On table 1 we summarize a possible set of information. Among all data available, some are truly important to the interaction, like Alice's mood, the number of doors on the room, just to point out a few. But is Bob's clothing relevant for instance? Let's study another, slightly different, interaction.

- **Situation 1b**, Alice asks Bob to adjust the air-conditioning temperature level.

Situation 1b requires a completely different set of information, for example Alice's clothing. The room's color plays no part on it. A limitation found in Dey's definition consists on neglecting what is considered contextual information (or simply entity's information) and context per se. In a more formal way, we have:

> *Context is relevant information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. Relevant information means information that, given a specific interaction between a user and an application, is useful and of meaning to that interaction and intervening actors.*

The main difference consists in distinguishing information and context. Although an entity can be described in different levels, not all information is useful on a specific interaction.

There are immediate consequences in system performance. Processing a complete set of information opposed to a sub-set of said information. Although an overhead exists due to the distinction that has to be made in order to only use *relevant* information, the end-result can be more efficient.

Context Representation

Several possibilities exist for context representation, as described in table 2, key-value models for instance are clearly limited, influencing information richness and quality, failing to address ambiguity, requiring that each system knows about the key-value model and how to use it. Partial validation should be highly desirable[19].

As for mark-up scheme models, the partial validation represents a strong concern. Validation is possible through schemas and validation tools, such as XSD or DTD, but disambiguation of context information must be done at application level. A comprehensive scheme definition is a step towards a high level formalism and thus may be used to determine interoperability[19].

The graphical approach comprehends the structure of the contextual information, facilitating applicability, although merging different models represents a limitation to some extent. Such models are mainly used for human interpretation of the problem.

Object-oriented modelling approaches are capable of a high interoperability, and inherent to the Object-Oriented paradigm, extension can be handled in a linear way.

Logic based models are very difficult to maintain, due to their extremely high level of formalism and applicability can be considered as a major drawback since in ubiquitous computing, logic reasoners are usually not available.

Ontologies can address most of interoperability issues. Validation (partial or total), formality, unambiguity and applicability make ontologies the most promising context representation technique.

In table 3 we show a number of context-aware applications and their associated context representation.

Proposed Representation

One important dimension of this project is interoperability between systems, as such, the context representation should be based on a standardised representation, such as OWL[1] and PIDF[2].

This dimension, along side with representation expressiveness and extensibility eliminates on-the-fly some representation techniques, namely key-value models, graphical models and logic-based models.

Being context highly dependent on the given interaction (see 2.1) extensibility must also be assured. Therefore, RPID[3] an extension to PIDF provides us with a standard representation of context/presence information which can be easily created, validated, shared and extended.

Context reasoning

When a computer system knows something (often a lot) about a user, it might have enough information to take actions on behalf of that user. When should an application take such actions? We will analyse different approaches to reasoning with different levels of "intrusion".

Passive context-awareness

There are genuine risks involved in allowing the system to take the initiative in any activity in which human participants are involved [17]. As we see, there is simply too much variability as to what should be done, given different conditions, for designers to successfully model appropriate outcomes in advance. This may not be problematic when the output of the application simply represents the location of a sensed object. Things get much more complicated when applications take it upon themselves to interpret human activity and "do things" for people by invoking services[4].

[1] OWL Web Ontology Language: http://www.w3.org/TR/owl-features/

[2] Presence Information Data Format - RFC 3863: http://www.ietf.org/rfc/rfc3863.txt

[3] Rich Presence Extensions to the Presence Information Data Format - RFC 4480: http://www.ietf.org/rfc/rfc4480.txt

	Alice	Bob	Door	Room
Context Information	Clothing Mood Age Etc.	Clothing Mood Age Etc.	Size Color Mechanical/Automatic Etc.	Color Temperature Number of doors Etc.

Table 1: Context information based on Dey's definition

	Complexity	Effort required to extend	Interoperability	Computer Understandable
Key-Value	Low	High	None	Yes
Mark-up	Median	Low	Total	Yes
Graphical	Median	Low	Total	No
Object-Oriented	Median	Low	Partial	Yes
Logic-based	Very High	Very High	Unknown	Yes
Ontology	High	Median	Total	Yes

Table 2: Context Representation models analysis

Bellotti et al.[3] claim that user intervention must always be required in order to a context-aware system to be accepted[4]. Through a series of experiments, Barkhuus et al. studied human reactions facing different context-aware systems.

Passive context-awareness addresses Bellotti's human concerns, allowing applications to directly gather and reason about contextual information but, rather that automatically and dynamically taking an action[7], they simply suggest that action.

Active context-awareness

Gu et al.[7] on the Service-Oriented Context-Aware Middleware (SOCAM) consider context-aware services as agents (hopefully through standards, such the ones defined by FIPA[4], and/or adequate languages, such as KQML[5]), applications and services that make use of different levels of context in order to adapt the way they behave according to the current context. This process does not take into account direct user intervention aiming for highly adaptable services during their life-cycle.

It is undeniable the diminishing of user control through this approach and users do feel that decrease, but still, most users prefer computers to take action on their behalf, after a suitable learning period adequate to the application purpose[3].

In our work we intend to composite passive and active context-awareness, providing services capable of automatically taking actions but based on user defined rules.

User-defined context-awareness

Another possibility of reasoning about contextual information is simply based on learning patterns.

After a learning period the system might know enough in order to provide the user with relevant information. Google Inc., the famous search engine, analysis search patterns and provide advertisement links accordingly. Analogously, Kawsar et al. use everyday objects, which maintain their original capabilities, to obtain the users context. Based on user pre-defined information (profession, interests, etc.) the *AwareMirror* can provide to the user relevant information (news, traffic conditions) on an usual object (mirror) while he/she is brushing his/her teeth (toothbrush is moving)[10].

Common social group's applications, such as MSN®[6], empower users with explicit context definition capabilities. Based on this attributions the application adapts (though slightly) its behaviour: a user's context of "Busy" avoids sound playing from the application.

IRC[7] allows for the definition of "away messages", providing additional information to possible incoming communications.

2.2 Social Groups

It is undeniable the quantity of social group's services that exist today (AIM, IRQ, MSN, GTalk, MSN, IRC, etc.), being evident that we couldn't interact with every single one (due to time and relevance restrictions), however one service would not be enough either. Extensibility and scalability are two very important conditionals that rule our system. This conditionals were indeed met and will be discussed further on.

GTalk

GTalk®is Google's approach to instant messaging. one of the main features about this service is the capability of interacting with it through several different means, from web-based applets to standalone desktop

[4]Foundation for Intelligent Physical Agents: www.fipa.org

[5]Knowledge Query and Manipulation Language: www.cs.umbc.edu/kqml/kqmlspec/spec.html

[6]Microsoft® Network

[7]Internet Relay Chat

	Key-value	Mark-up	Graphical Graphical	Object Oriented	Logic-based	Ontology Ontology
Bucca et al.[13]					X	
Capeus[18]	X					
Centaurus[9]		X				
Eclipse (plugin) [1]			X			
Hydrogen[8]				X		
Jini[12]	X					
OpenSocial [2]						X
SLP[12]	X					
StarUml [3]			X			
Stick-e[5]		X				

Table 3: Context Representation models and client applications analysis

applications, cellphone widgets or even email messages.

Since it uses XMPP (see section **??**) it is even possible for a person to develop their own application to interact with the service.

Sapo Messenger

Like GTalk®, Sapo Messenger®uses the XMPP protocol, meaning that it is not required to use the Sapo Messenger application to interact with the service.

One interesting feature however about their application is the capability of using one application to interact with both Sapo Messenger and MSN.

MSN

MSN®is without a doubt is the most used social group and instant messaging service. Although as mentioned above their protocol is proprietary, there are ways of interacting with this service that we explored.

Discussion

Social groups are roughly based on the same architecture. Although protocols might vary from service to service, the set of information provided by this services is the same: address information (communication end-point), and context information (presence).

Address information is not as rich as a address book analogous information, it provides however a communication end-point to the user. On the other hand, context provides (with its associated confidence) a new source of information regarding the user's availability or even activity.

All analysed social group services (MSN, GTalk and Sapo Messenger) represent real world networks. We have assessed the protocols that they use along side the importance that each one represents to this project.

2.3 Extract, Transform, and Load (ETL) Information

We will analyse the object identification problem, specially duplicate detection, addressing in turn conflict resolution.

2.3.1 Object Identification

Merging data from different sources requires that different representations of the same real world object be identified as such [11]. This process is called object identification. Object identification is difficult, because the available knowledge about the objects under consideration may be incomplete, inconsistent, and sparse. A particular problem occurs if no natural identifiers (IDs) exist. For instance, the URL of a Web page is a natural ID for the page. A meta-search engine can use the URL of reported hits to find and integrate duplicates. On the other hand, a used car typically has no natural ID other than the registered license plate for instance and sources about used cars do not store an ID. An integrated information system for used cars has no easy way of identifying a specific car being advertised in different data sources.

Object identification in the absence of IDs, which is essentially the same problem as duplicate detection, record linkage, or object fusion [16][15], is typically approached by statistical methods. Our approach is no different.

2.3.2 Conflict Resolution

Once different tuples have been identified as representing the same real world object, the data from them can be merged [14]. In general, a result that is integrated from tuples of different sources, contains tuples where:

1. The value for some attribute is not provided by any of the sources. Sources may not provide the value, because they do not store the particular attribute, or because they have stored a *null* value for the particular tuple. Because none of the

sources provide a value, the tuple in the result has no value either (*null* value).

2. The value for some attribute is provided by exactly one source. In this case, there is also no actual data conflict. When constructing the result, the single attribute value can be used for the result tuple. If a missing value has the meaning "not applicable" instead of "unknown", the absence of data can be taken into account as well. For the remainder, we assume the "unknown" semantics for null values.

3. The value for some attribute is provided by more than one source. This case demands special attention, because several sources compete in filling the result tuple with an attribute value. If all sources provide the same value, that value can be used in the result. If the values differ, there is a data conflict and a resolution function must determine what value shall appear in the result table.

3 Architecture

We want our system to be scalable, enhancing the richness of information that can be used, achieving that by providing an easy way to extend information sources.

To provide a scalable infrastructure the system was thought as a modular application. Each module has its own responsibilities and implementation, providing however a public, well-defined, generic interface that allows external applications to access its information and functionality.

In our exposition, specific vocabulary must be considered:

- *Interfaces*: Particular virtual classes that define the methods (and their signatures) that any class implementing such interface must provide.
- *Components*: Programming logic which provides or consumes a set of services. Components are the modules of an higher level application. Although components represent modules, and implement a generic interface, their inner workings might require direct interaction with the "outside" environment, being responsible of transportation of any ingoing or outgoing package.

During our analysis we found that our system requires two types of information, one regarding context information, the other regarding contact information.

Following this line of thought our system was basically divided in two parts, each one addressing each type of information.

The main component of our system, the *Group Enabler*, consumes contact and context sources in order to provide its own set of features. The number of each type's sources should not be limited, hence we used interfaces to catalyse scalability across the whole system.

As we can also see in figure 1, each external system has its own connector. This connector works as an adaptation layer between said system and our own. This approach allows for a standardized processing of information within our system.

System Entities

As any slightly complex system different concepts are involved. Furthermore the relationships between this concepts represent the core guidelines of any project's design (see figure 2). The core concepts to take into considerations are:

- **Group**: A set of group elements brought together based on a specific criteria.
- **Group Element**: A member of a group strongly connected to an entity, containing the mentioned entity's context (either *external* to the system or *internal*) and contact addresses.
- **Customer**: A specialization of an entity which represents a user of the developed system, having privileges to execute the system's features.
- **External Entity**: A specialization of an entity which represents an external user for our system. Although our system might have information regarding this entity (including context), that entity might not be aware of that.

4 Implementation

In this chapter we will analyse implementation specific situations and problems that arose during the development of the project. Since this project was developed under the standards and objectives of a communications company, several implementation aspects had to address company specific concerns and rules.

We will begin by focusing on social groups and how it is possible (and relevant) to obtain and use information provided by this type of systems. A technology discussion will address Enterprise JavaBeans (EJB), the relevance of this programming framework, and specific considerations regarding the Funambol system.

4.1 Social groups

Social groups are an undeniable rich source of information, but we also aim to go beyond the speculation and theoretical levels, hence we focused on interacting with real world, well-known services that not only provide

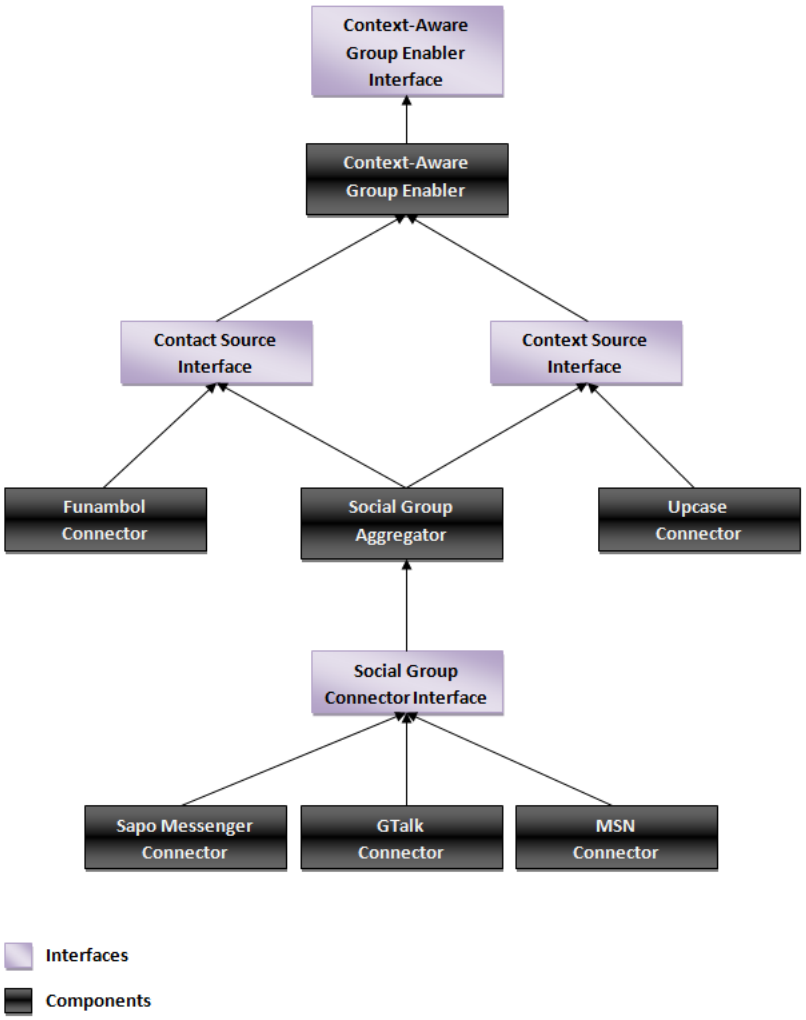

Figure 1: Solution's Architecture

the information we need but also adds to our application a real world value and a tangible objective towards commercial usage.

We will focus now on how does our system use these groups to acquire the said value. Roughly the process can be described in two stages, obtaining information and using information.

4.1.1 Obtain Information

In order for our system to obtain information from social groups it must have a way of representing the user before these groups.

The user must provide their authentication information to our system, which triggers a well-known issue, trust. Although our system does not use personal information beyond the scope of its features many users feel reluctant to share personal information with third-party applications.

In a scenario where the user provides his/her authentication information our system in term accesses the services on behalf of that user. On systems that allow multiple endpoints to coexist simultaneously our system is invisible. This approach prevents the occurrence of misunderstandings regarding the users actual location and availability.

Our system follows all steps that a normal client-side application does (see section **??**) not allowing however to communicate directly with anyone through it.

As any client-side application our system will be aware of the user's contact list, since the service itself provides this information.

The initial information, both address and context, is automatically gathered after successfully impersonating the owner. We however acknowledge that changes can occur quite frequently, specially information related to presence, and therefore context.

Virtually all social groups are based on the publisher-subscriber design pattern and most of them work on a push-based model (elaborate here or above?) to propagate this changes. This model assumes the usage of some type of listener to serve as a proxy of the subscriber.

4.1.2 Use Information

With the address and context information related to social groups gathered it is possible now to use

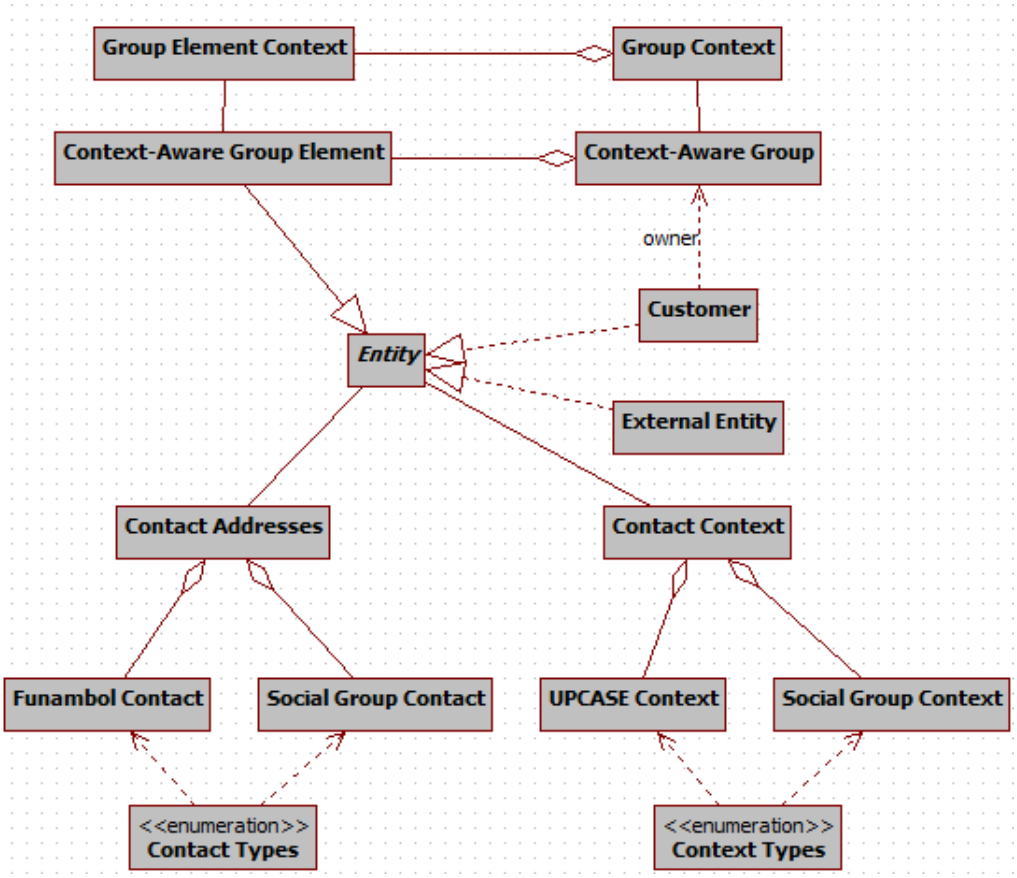

Figure 2: Information Architecture

it as necessary. This low level information must be converted to an agreed generic representation used throughout the system (see section **??**).

In the last section (section 4.1) we discussed the importance of keeping the low-level context information as is, but one important aspect also is to be aware if the aggregated information was provided by one or several different sources.

Any object representing social group context has by default an attribute which indicates from where the information was gathered. In the event that it originated on more than one source a "generic" label is applied.

4.2 Technology-related Aspects

All these techniques have a technological background, being decisions based on future work within the company's philosophy, compliant with existing and operating software infrastructures and technology choices.

For interoperability purposes the programming language used was Java. Java is a object-oriented programming language developed and supported by Sun Microsystems. Its execution is based on pre-compiled source code (known as byte codes) that run in any Java Virtual Machine (JVM), regardless of the computer's architecture. This disassociation with a computer's architecture and even operating system allows the usage of the same program anywhere with virtually no changes required.

Within the Java technology several different ways of development can be taken, from web development, applets, widgets, to enterprise applications. In our case we developed our system with the objective of an enterprise usage. To develop at this level Java provides the so called Enterprise JavaBeans Technology (EJB).

Component Organization (EJB)

Enterprise JavaBeans (EJB) technology enables rapid and simplified development of distributed, transactional, secure and portable applications.

Similarly to JSLEE specifications, EJBs are no more than building blocks that can be developed independently or can be combined to provide higher level features.

Having modularity, extensibility and scalability as main conditionals of our system (see section 3) the EJB technology provides an adequate development environment.

Our system uses Enterprise Java Beans (EJBs) as building blocks, most of which are stateless. This means that upon a request a bean is created to address it and upon completion it is eliminated. Lazy fetching the information is not possible in this situation since the objects themselves could (and most likely will) be

```
rule "Under Age Rule"
        when
                person : Person( )
                eval(person.age < 18)
        then
            System.out.println(pessoa.name +
            " is under age");
end
```

Listing 1: Rule Example

treated in another EJB or application.

EJB technology only addresses the development of building blocks, in turn these building blocks must run on an application server that provides all the technology's features and execution environment.

Once again this choice was made given the company's policy. The JBoss application server, an implementation of Java Enterprise Edition (J2EE) developed by Red Hat, was used to accommodate our EJB building blocks

Rule-based Inference and Rule Management

Rule-based systems satisfy the project's needs regarding context inference, which we built into our own system in order to aid it while reasoning.

Several implementations are available as of today, providing all sets of features related to this matter. We will look now more deeply into the system used during the project development, namely *Drools Rule Engine*.

Drools Rule Engine is a rule-based system developed in Java, it is JSR94 compliant[8] and aims for an open-source rule engine.

The system inner workings are beyond the scope of this document, we will however focus on rules themselves.

Whilst considering the possibility of using a rule-based system to infer higher level context an issue arose. What should be associated with rules? The whole system should not be bounded by the same rules across it nor an administrator should be restricted to use the same custom rules for all their owned groups. Rules are case-related hence each set of rules is associated with a group or an element group individually rather that being generic to all system.

The basic inner workings of a rule (see listing 1) are rather simple: for every object of type "Person" present in the rule-engine working memory it fires the "Under Age Rule" rule. If all conditions present in the "when" section are met then any action present in the "then" block will be triggered.

[8] JSR94: http://jcp.org/en/jsr/detail?id=94

5 Evaluation

In this section we will provide evaluation data regarding the developed system. Several dimensions must be considered and the tests undertaken simulate different usage patterns and data intensity. This project was developed under the scope of PT Inovação®company. The entire project is part of a commercial solution under development by this company, being as so an important asset for the overhaul capabilities and flexibility of the final system.

The measures obtained during the evaluation phase addressed different dimensions:

- **Source multiplicity**: determines if the number of sources has a relevant impact in the system.
- **Loading time**: determines the time required for the system to be fully functional and ready.
- **Contact sources processing time**: analyses the cost of iterating through all contact sources.
- **Context gathering**: determines the required time to obtain contextual information for all entities that need so.
- **Entries scalability**: determines how the system behaves given a raising number of contact entries.
- **Duplicate detection performance**: Although the algorithms are rather simple, the growth of entries might affect this dimension.
- **Database impact**: determines the impact of database persistence.

Test Environment

All tests ran on a Core™2 Duo CPU computer at 2.66 GHz frequency, with 4 GB ram memory and Windows Vista™Business 32-bit operation system. The tests were based on three prototypical test users with different characteristics:

The test analysis tables are now presented, upon which we will draw the evaluation results, but before analysing the statistical tables it is important to understand the difference, in terms of percentage, between the test users.

- Test user 2 has 350% more contacts than test user 1
- Test user 3 has 88.8% more contacts than test user 2

Loading

Analysing all test user tables (tables 4 and 5) we acknowledge that the loading time is constant, regardless of the number of contacts. The system was built having in mind scalability and extensibility, as such the connectors to be loaded should not limit the system.

	Average	Standard Deviation	Variance	X-min	X-max	% of total time	% increase
Load time	0.0568	0.028891	0.000835	0.026002	0.087598	0.5	17.8
Processing Contact Source time	3.4636	0.156292	0.024427	3.296992	3.630208	27.6	13.3
Total time to gather context	0.0102	0.009284	8.62E-05	0.000303	0.020097	0.1	8.5
Total time to persist entries	6.4996	0.425991	0.181468	6.045494	6.953706	51.9	44.7
Possible matches detection time	0.0926	0.02851	0.000813	0.062209	0.122991	0.7	1087.2
Group persistence time	0.3024	0.021149	0.000447	0.279855	0.324945	2.4	169.5
Useless	2.019	0.005196	0.000027	2.013461	2.024539	16.1	0.6
Total	12.5342	0.529033	0.279876	11.97025	13.09815	100.0	27.8

Table 4: Test user 2 - Results

	Average	Standard Deviation	Variance	X-min	X-max	% of total time	% increase
Load time	0.0548	0.024763	0.000613	0.028403	0.081197	0.4	-3.5
Processing Contact Source time	3.2308	0.266993	0.071285	2.946186	3.515414	23.1	-6.7
Total time to gather context	0.0264	0.021893	0.000479	0.003062	0.049738	0.2	158.8
Total time to persist entries	7.719	0.10464	0.010949	7.607454	7.830546	55.3	18.8
Possible matches detection time	0.3102	0.020389	0.000416	0.288466	0.331934	2.2	235.0
Group persistence time	0.5122	0.024601	0.000605	0.485976	0.538424	3.7	69.4
Useless	2.0362	0.005357	2.87E-05	2.030489	2.041911	14.6	0.9
Total	13.9632	0.283642	0.080453	13.66084	14.26556	100.0	11.4

Table 5: Test user 3 - Results

Furthermore the loading time represents a very small fraction of the execution time, roughly 0.5% of the total time.

Contact Sources Processing

The iteration through all contact sources is a very important aspect to be taken into account. Although it represents solely the iteration through the entries of a contact least, its performance affects significantly and directly the overhaul system performance.

As we can analyse the processing contact sources grows gracefully as contact entries grow. The major difference detected was when there were a 350% increase in the number of contacts (table 4) which increased this processing by 13.3%.

This graceful growth is very successful given the global impact of this dimension, which represents more that one quarter of the total time.

Group Persistence

It is important to distinguish between the persistence of raw auxiliary data from the processed refined group persistence.

This persistence has a higher increase rate compared to the previous persistence section. Its global impact of the former is much lower than the latter, but using test user 2 results (table 4), for a 350% increase in contact entries, connectors persistence grew 44.7%. For the same user, the group persistence grew 169.5%. Both types of persistence are below the contact growth percentage but they do have an unavoidable impact on the system.

Duplicate Detection

Duplicate detection, accordingly to the test results, must be reformulated in future work. The approach taken in this project revealed to be inefficient.

For a 350% contact entries growth (table 4) duplicate detection grew 1087.2%, similarly for a 88.8% contact entries growth (table 5) the same dimension grew 235%. This is the only dimension that performs worst than linear growth. Mathematical analysis demonstrates that for a contact entry's growth of N, duplicate detection grows roughly $3N$. Even though in complexity calculus constants can be neglected, when the value N tends to grow significantly, this ratio must be addressed.

Duplicate detection represents the current bottleneck of the developed system. For the tests undertaken its global impact is small, however its growth rate is reason for concerns.

Global Analysis

The developed system proved to be scalable. Given a high rate of growth in the number of contact entries, the overhaul performance of the system grew

	Tester1	Tester2	Tester3
Number of Contacts	2	9	17

Table 6: Test users characteristics

gracefully. The addition of new sources, either address sources or context sources, produce a minor impact in the mentioned system.

Stateless Java Beans allowed for on-the-fly resource allocation to take place.

Extensibility was achieved as well. The addition of new sources (GTalk®) required close to no effort aside from the interface implementation.

The interface approach was capable to provide a real extensible framework.

The system was ported to another environment, based on Linux operating system and PostgreSQL database management system. It was not possible due to company policies to perform tests in such environment. It is possible however to assume that the system is in fact portable.

Choosing the Java language and Hibernate as the persistence manager empowered our system with portability across platforms with very little effort required.

6 Conclusions

During the development of this project we concluded that the need for a context-aware application in the addressed scenario was real. The application should be capable of adapting its behaviour autonomously and dynamically according to external factors.

We analysed several existing approaches to context-aware software, either in terms of representation, reasoning, and sensing. Social groups are an important component of this project. Their information value (and commercial usage) meant a challenge and an asset. Ensuring interoperability between heterogeneous systems required the definition of capable data representation formats and translators. The information provided by these systems proved to be an asset based on the information richness gathered through them.

Information merging proved to be a complex subject, specially when customisation must be supported. Although the performance regarding the duplicate detection in particular was not as initially expected, adaptability is still a reality, with users being able to change merging rules as they see fit.

The architectural design of this project proved to fit the objectives pre-determined. The system's scalability and portability were observed, being interoperability already proved, due to the interaction with external systems.

References

[1] Matthias Baldauf. A survey on context-aware systems. *Int. J. Ad Hoc and Ubiquitous Computing*, 2, 2007.

[2] Louise Barkhuus. Context information vs. sensor information: A model for categorizing context in context-aware mobile computing. *Symposium on Collaborative Technologies and Systems*, pages 127–133, 2003.

[3] Louise Barkhuus and Anind Dey. Is context-aware computing taking control away from the user? three levels of interactivity examined. *Proceedings of UbiComp*, pages 150–156, 2003.

[4] Victoria Bellotti and Keith Edwards. Intelligibility and accountability: Human considerations in context-aware systems. *HUMAN-COMPUTER INTERACTION*, 16:193–212, 2001.

[5] P. J. Brown. The stick-e document: a framework for creating context-aware applications. *ELECTRONIC PUBLISHING-CHICHESTER*, 1995.

[6] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 2001.

[7] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 2005.

[8] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. Context-awareness on mobile devices - the hydrogen approach. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*, page 292.1, Washington, DC, USA, 2003. IEEE Computer Society.

[9] Lalana Kagal, Vlad Korolev, Harry Chen, Anupam Joshi, and Timothy Finin. Centaurus: A framework for intelligent services in a mobile environment. In *In Proceedings of the International Workshop on Smart Appliances and Wearable Computing (IWSAWC*, 2001.

[10] Fahim Kawsar, Kaori Fujinami, and Tatsuo Nakajima. Experiences with developing context-aware applications with augmented artefacts. *Experiences with Developing Context-Aware Applications with Augmented Artefacts*, 2005.

[11] William Kent. The breakdown of the information model in multi-database systems. *SIGMOD Rec.*, 20(4):10–15, 1991.

[12] Ludwig and Maximilians. *Service Interoperability in Ubiquitous Computing Environments*. PhD thesis, University Munich, 2003.

[13] John McCarthy and Sasa Buvac. Formalizing context (expanded notes). Technical report, Stanford, CA, USA, 1994.

[14] Felix Naumann and Matthias Häussler. Declarative data merging with conflict resolution. In *International Conference on Information Quality (IQ 2002). 2002*, pages 212–224, 2002.

[15] H. B. Newcombe. *Handbook of record linkage: methods for health and statistical studies, administration, and business*. Oxford University Press, Inc., New York, NY, USA, 1988.

[16] Yannis Papakonstantinou, Serge Abiteboul, and Hector Garcia-molina. Object fusion in mediator systems. pages 413–424, 1996.

[17] Gene I. Rochlin. *Trapped in the Net: The Unanticipated Consequences of Computerization*. Princeton University Press, Princeton, NJ, 1997.

[18] Michael Samulowitz, Florian Michahelles, and Claudia Linnhoff-Popien. Capeus: An architecture for context-aware selection and execution of services. In *Proceedings of the IFIP TC6 / WG6.1 Third International Working Conference on New Developments in Distributed Applications and Interoperable Systems*, pages 23–40, Deventer, The Netherlands, The Netherlands, 2001. Kluwer, B.V.

[19] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.