A Personalised Ontology and Rule-based Approach to Managing Message Overload

A thesis submitted to the
University of Dublin, Trinity College,
for the degree of
Master of Science in Computer Science

Fengjiao Lv

School of Computer Science & Statistics,
Trinity College, University of Dublin,
Ireland.

2015

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

_____

Fengjiao Lv

August 28, 2015

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

_____

Fengjiao Lv

August 28, 2015

Acknowledgements

I would like to gratefully and sincerely thank my supervisors Prof. Owen Conlan for his guidance and encouragement. I have learned many things from him. He spent much time guiding me how to write a paper, search literature, and do the poster carefully and patiently. Also every week he gave many useful suggestions and shared interesting, creative aspects of this research topic and research life. I would like to thank Sara Alzhrani, with whom I worked closely and we puzzled over many similar problems. Most importantly, we developed friendship during the master semester study at Trinity. I would like to thank my roommate, Elena Lee, a South Korean girl. We shared many pains and happiness together, gradually becoming independent and confident, which made every normal day in a different country meaningful.

Abstract

Large numbers of messages keep distracting people from current tasks and interests. Also, messages take up valuable resources of devices and deteriorate performance. This thesis aims to explore the combination of rule-based approach and personal ontology to mitigate these issues.

Two types of information are identified: one from messages, user profiles and context. This information is semantically uplifted into ontological instances for further processing. However, it poses a number of issues as messages are dynamic, and context and user interests can be fickle. These problems will be mitigated by proposing personal ontologies. The other information is from domain knowledge, intended to form into rules for logical reasoning and taking actions.

Two use cases are created that contain interrelated messages, dynamic context, and interrelated persons, etc. Two corresponding research objectives in the thesis are 1) exploring the effectiveness of rule-based approach, by comparing rule languages and employing complex rules derived from use cases; 2) combining static keyword lists and dynamic personal ontologies, to semantically match message-message, message-context, and message-interest.

In response to novel messages and novel senders without historical reference, the thesis designed rules to automatically learn and test the result of this rule-based approach. Another key challenge of the system is how to deal with large numbers of messages. The thesis proposed a chaining, distributed approach to making the phased operational decision on pending messages. Another challenge is how users can give feedback and involve into scrutiny. To meet the requirement of scrutability, the system asserted rules to changing the level of user interest. The current approach combined the research in the area of semantic uplift, ontology modelling, personalisation, rule-based system, etc. to face the addressed challenges. Afterwards, the application are evaluated iteratively to validate the goals of the research.

An experimental rule system called message assistant is designed and implemented with contextual information, and user interests. A Java message simulation was implemented to generate controllable numbers of metadata messages of selected time period as required.

In summary, from the system's point view, messages are handled with a dynamic personalised ontology, chaining rules, keyword lists. Such combination are detailed illustrated in following sections. From a research view, rules transform semantic information, determine operations at differing stages, however, rule-based approach has limitations in the situation of limited knowledge, which will be discussed in the future work.

Table of Contents

List of Tables

List of Figures

Chapter-1 INTRODUCTION

1.1 MOTIVATION

Diverse messages are pushed to people at any time, no matter what they are doing. Not only does it distract people from current tasks but also takes up valuable resources of devices and deteriorate performance. It stimulates the demand to research an intelligent system to deliver urgent messages, or messages of use interests to people immediately, send less important ones later and delete spams, acting like a message assistant.

Personalisation and adaptivity technologies are novel and efficient. Past applications and research put much effort on filtering messages depending on messages and applying content-awareness rules. Alternatively, some traditional filtering methodologies, like collaborative filtering, attempt to improve user experience by exploring common preferences among many users. Currently, research surrounding personalisation and context-awareness prove problematic and difficult, because context, user interests and messages are dynamic. It is required to purpose ontologies as a means of achieving dynamic precise matching using rules and ontologies.

The rule-based approach has been used as a novel way to manipulate domain knowledge to leverage information. It has been used in business areas, artificial intelligence, semantic web and adaptive computing. As surveys shows [22, 23], many start-up companies adopt rule-based approaches as rule-based applications are lightweight and flexible. Rule-based system, using script language and easy to transplant, displays many benefits, but it has many limitations. Moreover, different rules engine has different working process and features. It makes necessary to compare rule engines and rule-system architectures, explore the efficiency and effectiveness of rules and rule-based system.

The thesis aims to explore rule-based approaches and personalised ontologies which integrate user interests and dynamic user current context. In addition, this thesis compares the completeness and reasoning capability of current technologies for representing information, the efficiency and effectiveness of rule languages and rule engine, as well as architectures.

1.2 RESEARCH SIGNIFICANT AND OBJECTIVES

The research aims to provide and evaluate a personal and adaptive approach based on rules in handling messages intelligently.

Research Significant:

- Compare technologies in information representation.
- Investigate applications of ontologies in semantically bridging the gap between ever shifting user interests and messages.
- Compare rule language, rule engine and rule-based system.
- Analyse architectures to handle large numbers of messages and increasing numbers of rules.
- Explore the effectiveness of rules, especially combining complex context, and dynamic user interests.
- Evaluate rule-based applications to explore the feasibility and flexibility of rules, especially when there is limited knowledge base, using an iterative method in development and control-variable method in experimentation.

Five objectives are listed here:

Objective 1: Find out similar approaches to categorising and operating messages from the state of the art, and investigating issues surrounding rule-handling approach, and ontology modelling, etc.

Objective 2: Design and implement the information representation based on two stated two use cases, and evaluate this approach.

Objective 3: Design and implement the rule-based approach to handling incoming messages based on the two use cases. Then, evaluation and comparison based on the current approaches should be given.

Objective 4: In order to face the issue of massive messages, design and improve the general architecture, and evaluate the architecture.

Objective 5: Follow an iterative approach to evaluating and validating the design and implementation of the design and implementation.

1.3 RESEARCH QUESTION

The system comprises three key components; they are metadata (consisting messages, context and user interests), ontologies and rules. In order to produce dynamic personalised and rule-based system, a number of issues are posed when reviewing the state of the art.

Here are some potential options before the research: in which method can system format information into metadata precisely; how to achieve precision between user interests and messages; how to classify and markup individual messages using ontology reasoning; how to

build adaptive ontologies in handling dynamic information to facilitate personalisation; how to scale system to deal with large numbers; how to choose and design rules; how to visualise user scrutiny.

Many research issues as listed above can be delved. However, in this thesis, only some vital are selected and put into priority.

Major Research questions:

- To what extent can a rule-based approach that also integrates a context model and personal ontologies be leveraged to manage incoming messages
- Evaluate this high-level framework.

Minor Research questions:

- Propose applications of ontologies to match dynamic messages and changing context, messages and interests, in order to let users be presented with either urgent or interested messages immediately.

1.4 RESEARCH PROCESS AND METHODOLOGY

Centred on research focus and research objectives, the process started with surveys on current research and applications, and comparing strengths and limitations of these technologies from theoretical perspective view. Key findings are concluded surrounding disordered rules and facts, personalised ontologies in handling dynamic metadata and context-awareness.

Specifically, a system was designed and implemented based on the rule and personal ontology approach, following an iterative methodology. The development was conducted with simple rules centred on use cases. With the increasing complexity of rules, the system proved to be feasible to make ontology reasoning even though some other ontology technologies like OWL exerted better capability in some scenarios.

Worth noting is that the thesis designed both a centralised system architecture and a distributed architecture. Specifically, a detailed analysis was made on both architectures, from functional analysis to performance, load-balance, security, etc.

In evaluation part, a three-stage experimentation was given. First stage centred on use case was to test the feasibility and performance of the test system; in second stage, to test the scalability of the system and effectiveness of separate rules, a message simulator generated controllable amounts of metadata messages. It used control variable methods to take experiment and EXCEL to make regression analysis; last stage focused on the capability of handling novel messages, with a message generator retrieving emails.

In summary, the research process is: theoretical comparison -> use case –> complex rules –> large numbers of messages-> novel messages.

Research methodologies are iterative development, control variable methods, and regression analysis.

1.5 CONTRIBUTION

Several contributions are identified.

Contribution One(Major): The thesis provides a novel, efficiency approach in handling messages; Message assistant system is a test-oriented, light-weight, efficient and intelligent one based a rule-based approach, integrating context, and user interest ontology.

Moreover, this approach contributes to the state of the art by offering a distributed architecture.

Contribution Two: Explore the feasibility, simplicity, scalability, and flexibility of a rule-based system. Design and implement a series of applications (Message simulator, message generator, etc.) to evaluate the effectiveness of such approach.

Contribution Three: Propose and realise personal ontologies as a meaning of achieving precision between dynamic messages and context, messages and interest, messages and messages.

Contribution Four: The system was using iterative development and three-stage experimentation, which carried on series of experimental and analytical tests on performance, scalability and load balance from perspectives of a system.

1.6 THESIS OVERVIEW

The thesis consists of seven chapters.

Chapter Two provides analysis on the start of the art. This part summarises current research and applications on rule-based approach, personal ontology. In order to develop feasible application, the start of the art also explores some other areas related to such topic, which aims to make system scale, increase accuracy of personalisation and every operation decision, tolerance fault even though there are limited knowledge available.

Chapter Three demonstrates two use cases as well as key challenges. Then it outlines the requirements for this research.

Chapter Four illustrates the design of such system. It begins with analysis on two possible architectures from the perspective of performance, security, feasibility and scalability. Then it describes a detailed design for ontological modelling and rule design to meet the requirements derived from Chapter Three.

Chapter Five describes an iterative implementation on rule-based approach. For the purpose of evaluation, this section starts with an implementation of message simulator to generate conditional messages. Worth-noting is that it implemented a simple OWL to compare Owl and ontology.

Chapter Six presents three-stage evaluations on both functional and non-functional experiments. Ranging from use cases to a large number of messages to novel messages, the rule-based approach presents its advantages while reveals its disadvantages.

Chapter Seven concludes this thesis. Future work of this system and main contributions are discussed at the end.

Chapter-2 STATE OF THE ART

The research focuses on message uplift and reasoning by using rule-based methods, integrating personalised information that meets user's interests and context[3], (a similar approach is raised is in [2]).

The main research objective is to explore the effectiveness of rule languages, rule engines under various situations: messages from heterogeneous sources, interrelated messages, or dynamic personality as well as changing context, novel messages.

Several surveys will also be reviewed to find solutions of the issues when handling a large volume of messages and novel messages, dynamic and complex context. More surveys will be analysed in terms of the scalability of architectures [4], adaptivity via machine learning and feedback [1, 5-9], system scrutability [6, 8, 10] as well as context awareness [11].

2.1 INTRODUCTION

According to the goals illustrated at the beginning part of the session, this session will begin with exploring the issues with addressing the principle of rules, comparing three formats of information representations and defining the context.

After the initial start of the art was concluded on the theoretical comparing in rule languages and information representations, more systems were investigated and given analysis of corresponding architectures, approaches to handle rules, personal ontologies and context ontologies. Then, this leads to extending the research on context-awareness integration, distributed system design in regards to massive messages, and user-involved scrutability. So more systems and research are carried out and analysed in order to solve these issues.

The literature review focusing on functional and non-functional comparison will assist in understanding of the challenges in this area and bringing out the strengths/weaknesses among existing systems and approaches.

2.2 INFORMATION REPRESENTATION

In the research, two kinds of information are identified: information from heterogeneous messages, dynamic contexts and user interests; information from domain knowledge which

formed into rules to take actions to messages. To represent information, semantic web techniques are adopted to generate both human and machine understand forms of information. Ontologies, RDF[12] and OWL[13] are suitable for the purpose of the research, but all of them have features, benefits and weaknesses.

Ontologies, frequently using extensible and markup language (XML) to model a domain in a structured, semantically rich way. It includes classes, inheritance, and relationships between classes and instances. The main benefit of ontologies is that it provides reasoning over relationship defined in the ontologies and connecting instance to abstract types. In this thesis, we could reason that Mary (an instance) is a Person (class) interested in music (an instance of class interest), Mary is a family of (a relationship) John.

RDF (Resource Description Framework) [12]. Given that metadata has the highly variable and highly extensible characters, RDF is recommended as a metadata model to describe and model concepts and their relationship on the web in a variety of syntax format.

The underlying structure in RDF is Subject-Predicate-Object triples. RDF statements can be displayed in a labelled, directed multi-graph, and is designed to describe knowledge, rather than data, in that it focuses more on data relationships and reasoning from data to date. Thus, an RDF-based metadata model suits describing knowledge than ontological models and relational models [12, 14].

However, the drawback of RDF is its informality; not only is the set of predicates defined in an inform way but also it is easy to pre-suppose human languages and some common understanding which is incorrect in different contexts. Another drawback is also about imprecise, which means some words are open to misinterpretation if it is defined without contexts. Currently, W3C adopt XML namespace (URL) to avoid ambiguity[15].

OWL (Web Ontology language) [13, 14, 16]extends many web standards like XML and RDF, and much more formally specified than RDF to represent knowledge of diverse domains and various.

OWL is built upon RDF, but provides a mechanism to process and reason via using formal semantics and RDF/XML serialisations. Moreover, it supports semantics in a well-defined way, formal properties which reducing the steps of decidability, known reasoning algorithms and many implemented systems.

Last but not least, OWL is able to cooperate with SPARQL [17] and other semantic web techniques. OWL is used to model human knowledge into ontologies, generating a set of ontologies classes, constraints and properties [14].

Conclusion:

Besides, the comparison among RDF, and OWL gives us a basic knowledge about the effectiveness of information representation. The triples of OWL language extends the capacity of information description and reasoning.

Even though some definition languages, such as OWL and RDF are able to present information, we still choose ontologies in this research because it is easy to maintain and lightweight ontologies make this feasible enough.

Related tools for ontologies.

Apache Jena [14] is a java system for manipulating RDF models, building semantic web applications. Jena includes many APIs to interact with persistence storage and process RDF data.

Protégé is developed in Stanford University. It is a free, open-source platform, an ontology editor. Not only does it have a graphic user interface to establish ontologies, but also provide reasoning and inference engine to validate the models and infer new information based on the user-defined rules and analysis of existing ontologies. Protégé is written in Java and has good compatibility with Java.

Protégé supports working with many rule languages in Protégé-OWL [18, 19].

SWRLTab is a development environment for defining and running SWRL rules which include sets of libraries to work with XML files, RDFS, etc.

JessTab [20] [21]is a Jess development environment to work with Jess. With JessTab, Protégé knowledge bases are directly managed by Jess rules and protégé also extend Mata classes to be mapped to facts.

Conclusion:

Ontologies are feasible enough to help solve research questions even though OWL has more advantages in describing the domain and reasoning. It is able and effective to use tools such as Protégé and Jena, to make manual maintenance and updating ontologies easily. When needed, users or system can add or edit classes, relationships and instances.

2.3 RULE AND RULE-BASED SYSTEM

2.3.1 RULE-BASED LANGUAGE AND RULE ENGINE

This session will compare some rule languages as well as effectiveness and limitations of these rules languages.

2.3.1.1 Rule Definition

Rules as defined in normal life describes effects which have an association with causes, actions, and conclusions [22-25]. Thus, rules are frequently leveraged to represent processes, logics and grammars. The rule engine is used to choose an appropriate rule in reacting to differing conditions in running cycle, which makes the system is flexible to interact with rule-based database system [26, 27]. Therefore, rules constitute a part of modules to compute and are aggregated with other rule bases and rule base modules.

Another worthy noting features of rules are that some special implications in rule bases has the same effect as taxonomic in ontologies; that is, rules provide ways to transform query and extend ontologies.[23]

2.3.1.2 Rule-based Language

Jess

Jess*[26, 27]* is a light rule engine for Java platform which can be run as a standalone program or embedded into java applications. In Jess, working memory stores data that can be transformed by dynamic rules. With Jess, Java software is equipped with the capacity to reason using knowledge in a form of declarative rules. Jess is an effective, light scripting language which is able to interact with all of java interfaces.

The features of Jess: access to all of java interface; Apply dynamic knowledge to data in working memories effectively; has consistency checking.

RuleML (The Rule Markup Language)

RuleML[28] is an XML-based rule language which spans across industrially relevant kinds of web rules. It acts as a bridge to connect comic logic. Therefore, RuleML is a translator as an interchange language; it realises internal mapping between sublanguages, presentation syntaxes, and external mapping between *RuleML* and other languages.

Semantic Web Rule Language (SWRL) and BaseVISor

OWL [13] The Web Ontology Language has the capability to realise automatically reasoning with the help of its rich classes and properties. However, OWL has the limitation in rule reasoning because of its limited expression when properties are composed of other properties.

SWRL[16, 29] is a rule languages combining OWL DL and OWL Lite sublanguages of the OWL with unary/binary datalog RuleML sublanguages of the RuleML (Rule Markup

Language), which extends OWL with Horn-like rules. The known limitation exists in that SWRL is undecidable and has few reasoning engines. In order to support complex logical conditions and a set of DL constructs, BaseVISor[30] is a rule language, based on Rete network to auto-process triples, using XML syntax to define facts. BaseVISor creates rules and issue queries including a forward-chaining inference engine to process RuleML rules and also it incorporates consistency rules. In the experiment, the BaseVISor is able to process over 300 classes and many properties within a few seconds. But it remains to be determined exactly the effectiveness of the intersectionof rules [28, 30, 31].

Comparing BaseVISor and Jess[16, 31]:

Jess has already existed for a couple of years. Many legacy codes exist in Jess codes to support LISP-like list structures that CLIPS supported. However, in semantic web, RDF/OWL only contains triples which made the codes redundant and increase the overhead. In comparison, BaseVISor only deals with triples, however, Jess works a lot with internals while BaseVISor prevents developers from going into more details of the internals because of its limited space.

2.3.2 RULE-BASED SYSTEM

Theoretical research forms the basis analysis to meet the research goals. This section will focus on current rules research and applications, giving readers the borders of rule-based systems. It highlights the importance of the research on rule-based approach. The fundamental rule-based approach can be applied to large numbers of other applications and areas.

From the author's research[22, 23], rule studies has three main interrelated branches: Business rules [36] and automatically processes, Rules and semantic web, personalization and rule-based event processing, learning. These focuses reflect the role of rules in many application areas: processing Semantic Web data, reasoning over actions and events as well as specifying automatically business process and policies. IBM, in particular, has contributed many pieces of research at this area.

The cutting-age research on rules include: 1) Pragmatic web, the next version of the web after semantic web, takes web as a platform to communicate and coordinate by putting context and rules together [37].2) mapping the relational database to knowledge bases, the challenge of this lies in using rules to map primary-keys, foreign-keys constricts to knowledge bases [38].3) translating Business Process Model Notation (BPMN), a business process model into semantics business rules, making it easy to understand, like graphical representation [36].

A trend for a rule-based system is real-time, heterogeneous, distributed using rules. As to real time, a production system using Event Calculus[39] presented in [40]is a forward-chaining rule-based system, adapting the declarative semantics of the logic knowledge, solving the problem of retrieving and reasoning knowledge about change and causality in dynamic domain . In regards of heterogeneities, given that service oriented architectures and event driven has become trend, the paper from [41] presented a semantic approach called Hiddent Algebra approach to realise mapping between RuleML and Behavioural Algebraic Specification Language, which provides better automated reasoning capabilities. From the paper, Rule works as a translator to overcome the problem of multiple languages and automate these processes. Similar to prolog language, the Event Calculus performs temporal prediction and monitoring using rules to join many paradigms. Rules are also helpful in agent processing domains. The paper from [42] developed DALI, a logic agent-oriented language that equipped with "event-condition-action" rules. DALI makes internal conditions as events which can be defined by the programmer, in this way, enabling choosing actions to undertake in response to complex preference and also based on sets of past events.

2.3.3 SUMMARY

In this part. The effectiveness of rule languages is compared as well as the efficiency of rule engines. The limitation in one rule-language can be the benefits of other languages. It suits to different situations. When going back to use case, 1) it is required to generate the reasoning process, Jess is better than RuleML and SWRL in that it goes into details of reasoning procedure, 2) it is required to category messages based on sender-receiver relationship, and message-relationship, SWRL and RuleML is better than Jess in that it adapts OWL and rich classes to describe and reason these relationships. To conclude, rules can extend knowledge, semantic information. But has different limitations.

2.4 CONTEXT

In order to meet the main research objective which applies dynamic rules to incoming messages, context should be briefly described here to form the basis for the analysis.
The definition of context from[32] is

"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. "

From the definition, it is apparent that context information is difficult to put into a general form and quantify. Given that current research focus on semantic web and knowledge bases, context information will be unified into RDF/OWL. Thus, the state of the art of context here focuses on ontological context.

The challenge of context information is to unify heterogeneous context information and combine them in a sharable, uniform format.

Many systems [2, 33-35] provide approaches to track heterogeneous context information into a unified format. Worth noting is that the thesis from O'Connor[11] mentioned a novel way to integrate contextual information. The thesis uses SWRL to establish common concepts in different ontologies. The framework developed a novel way to exchange information without creating an integrated model of context in advance. The enrichment of the target application to include both schematic knowledge enrichment. It adopts a rich, flexible representation to represent a relationship between separate concepts. In this way, the system represents good performance as the evaluation displayed. A context mediator is envisaged as being at the heart of an informing environment.

Conclusion:

The thesis provides good architecture as well as theoretical background to represent contextual information in an integrated way. Given that the research focus is not in context-integration, the thesis will not study this area deeply. However, the complexity of context and how to compose this information into a unified form pose many problems which will be discussed in future work of this thesis.

2.5 PERSONALITY

A Personalized Location Information System (PLIS) in [43] sends personalized, contextualized information to users with the help of rule-based policies. PLIS provides a connection between point of interest (POI) owners (for instance shops), and users. POI has general rule-based policies like some marketing strategy. User's preference and contextual information can be combined together depending on the rules fired. Another feature of the system lies that it is XSLT technology to transform RuleML format to Jess rules [43].

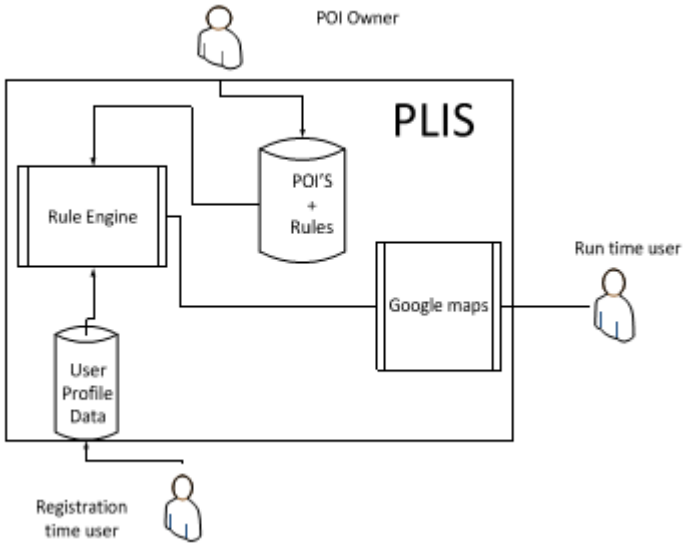


Figure 1: The Framework of PLIS

The Figure 1 shows the general design of the system. User data and Buiness data(POI) is collected by Jess Rule engines. Jess rule engine uses the asserted facts to update the POI attributes value depending on the user's context. At last, data is delivered to a user and performed for visualization.

Strengths and Weaknesses in the system

The system provides personalised and contextualized location information to users. Just as the experimental testing result shows this rule-based method enhance the power of displaying location information about shops.

However, the system can be improved in various ways. First, The POI and user information can be represented in RDF/OWL format for better flexibility and self-reasoning. Second, because the user interests are statically added at the beginning, it made the system hard to adapt to pragmatic situation that user interests can get transformed. It is recommended to consider user involvement in decision making or consider machine learning based on user feedback; last but not least, this centralized design fails to take scalability or increasing shops into account.

Conclusion

A rule-based system can offer a boost to the quality of required information by embedding user profiles, contextual information and business policies.

Adaptive locator system [3] is implemented which adopting ontology language and knowledge-driven filtering. This essay provides a well-defined framework for personalized

ontological reasoning. It presents a three-level abstract PERSONAF framework to meet key demands of ontological reasoning. The three levels are displayed here:

- Middle building ontology. Using the Web Ontology Language, OWL,8 using Protégé
- An application ontology represents a particular building
- The novel part of PECO is the accretion ontology

So the adaptive locator makes use of the ontology to deliver personalized information about the people in a smart building. In addition, the essay demonstrates the power of implemented framework and evaluates scrutability based on the user study.

Benefits and weaknesses

Three-layered framework, considering ontology and dynamic context is very clear structured. Other benefits of the system lie that it reduces the possibility of mistakes by taking account of user feedback and the decision is visible to users.

The weakness of the system is that it takes consideration limited number of information, and the reasoning is weak: it generates decision from many different methods and lets users choose, so it depends on the historic logs of choice.

Conclusion:

The three-layered architecture is reasonable in the dissertation, semantic uplift, ontologies (context, message and user interests).

The dissertation will adopt the rule-based expert system shell to enhance the ability to reason from ontologies.

As to personality, many similar systems [5, 43] purposed approaches or framework similar to the systems listed above.

2.6 INCREASING MESSAGES: PERFORMANCE VS. SCALABILITY

With the increase of messages and increasingly complex knowledge management, it is clear that single server is not enough to deal with large volume of data at the same time. In the part, some existing distributed message systems are analysed. It can also give some hints for future work.

Minerva [4] is a storage, inference and querying system for large-scale OWL ontologies by using relational databases. The benefits of the system lie that it meets the scalabilities requirements and also has high query performance. Minerva is developed by IBM integrated Ontology Development Toolkit (IODT). It combines DL reasoned for the TBox inference with

logic rules for inference. Then, mapping between describing logic and logic programs is enforced. The components of Minerva are Import Module which consists OWL parser and two translators; Inference Module which includes rule engine and DL reasoned; and Storage Module which stores inferred assertions by the DL reasoned and rule inference engine; Query module which is supported by SPARQL via directly retrieving from database using SQL statement. The high performance is owning to this part because the inference has already done at the time of loading data and no need to enforce inferring at this stage. It is expected to improve the response time.

2.7 USER SCRUTINY

User involvement can be implicit or explicit. For current system, it is of importance to design an adaptive system to meet user's current needs, knowledge and preference.

The mentioned system in [3, 7, 8, 44] has already taken implicit adaptive user involvement into account, by showing users the decision process and asking for user feedback. Some details about scrutability are derived from the systems listed below.

An item recommendation system [8] is illustrated in the paper, it combines content and critique-based meal recommendations with scrutability and user feedback. Weighting in the recommendation algorithm was heuristically derived.

Weakness:

It is a simple implementation system that adopts existing recommendation algorithm based on XML-content and give visible scrutability. But the system is fairly simple but it gives basic design thought.

The paper [6] defines the meaning of scrutability. In the paper, a system called PLUS designed an active user models which are able to react dynamic user Interests. In PLUS, the novel user will begin with user modelling processes from long term user models. With the system adaptive learning by collecting evidence, the user model will be built. PLUS is based on accretion/resolution approach to managing the long-term user models. In particular, accretion is responsible for evidence collection while the resolution will interpret evidence and confers much flexibility.

Benefits and weakness

The system explains scrutability in details from many aspects. Essential for adaptive scrutability is at all aspects. Another benefit of the system is that it adapts ontologies to explain to users about the component in the user models.

2.8 RULE-BASED SYSTEM VS. HEURISTIC-BASED SYSTEM

Human reasoning has been accounted into two complementary categories. That is, one is analytical, logical and reason oriented, cited as rule-based processing; the other is experimental, affective and associatistic. It is similar to decision-making in automatic computing system[25].

The paper [45]gives a detailed comparison between rule-based processing and heuristic-based processing. However, a rule-based system has its limitations. In some cases, when adapting rule-based, the long decision needs complex mathematic and logical reasoning but the rules to be selected is predetermined. Also, the effortful rule-based system is required to make a difficult decision when a novel event happens. The paper recommends integrating rule-based and heuristic-based processing.

Chapter-3 **REQUIREMENT**

3.1 INTRODUCTION

In order to explore the feasibility and effectiveness of the rule-based approach in information filtering and reasoning, a message assistant system is designed for later experimentations at this section.

The system is an experimental system to help users filter, uplift, classify and operate (delete, pending, deliver) incoming messages from messages and apps, based on user preferences and current context.

Two use cases are designed. Use case 1-john focus on context ontology, categorising messages based on content and handling group message. Use case 2- Anna centred on personal profiles, matching messages with user interests. The use cases contains increasingly complex scenarios, which the system needs implement complex rules. And many corresponding design schemes about ontology design and writing rules to implement the scenarios are raised. In illustrating use cases, system requirements are generated, ranging from simple ones to general cases with hundreds of rules and messages.

3.2 USE CASE

Use cases should be designed as complex as possible, which helps to explore the completeness of semantic ontologies and effectiveness of rule system.

Even though use cases simulate the particular example and have limited numbers of scenarios, it is able to extend to more general examples. The use cases give space to face the challenges of various messages by second, dynamic contexts by minute, changeable user preference by day.

3.2.1 USE CASE 1-JOHN

The first use case centres on white-collar people, who needs the system to help them improve work efficiency. People worked in the office need to keep focus.

- **General Description:**

John works as a HR. In daily life, he needs to attend meetings when he does not to receive unrelated messages, appoint with interviewees and co-works which he needs all these pending

messages about the person to be delivered immediately. During work, he needs to go to coffee shop to do some business. So for him, coffee shop can be a work context which needs to pay particular attention.

After work, John likes football and chats with friends in a coffee shop. He is free to be called by close friends, notified by apps and football events.

Attention: Some messages from diverse source **imply** the urgent of messages. So pending ones should be upgraded to delivered, deleted ones should be pended or delivered.

A coffee shop can be a sign of work location or a resting location, depending on the time.

The particular experiment for John in a day is illustrated here:

- **Summary:**

Context:

Work time: 1:00 p.m. - 3:00 p.m. he does not want to receive friend invitations as well as advertisements.

Free time: 3:00 p.m. - 5:00 p.m. he would like to receive advertisements messages about football.

Simulated Messages:

1) 1:00 email from Mary (wife) about banquet tonight at 9 p.m.

// pended until 3 p.m.

2) 1:10 text from a co-worker (Sue) about company news.

/delivered. From co-worker about company information

3) 2:00 email about urgent meeting (from secretory) right now for 20 min.

//deliver

4) 2:10 text from friends (Sunday afternoon).

//pending until 3 p.m.

5) 2:10 text from co-works about new work plan.

//pending until 2:20

6) 2:30 text from Mary about banquet tonight at 9 p.m.

//delivered. The message is upgraded

7) 3:10 Facebook about friend request.

//deleted

8) 3:20 Facebook about a football game.

//delivered, meet with user interests

9) 3:30 Facebook about music events

//pending.

In addition. At four o'clock, John checked the pending list, and he found that music events are in the state of pending. Actually, he likes it. So he upgraded the interest level about music and increases the keyword for music.

- **Rules**

Complex rules are distilled from use case 2:

I. Group of similar messages should be upgraded.

It is required that logical reasoning to identify the similarity of messages. The challenge exists in discovering the similarity of incoming messages, removing duplicates, and upgrade the messages.

II. Identify messages from family and messages from a single related person.

It is required that the system can identify the categories of the incoming messages from the sender information.

III. Categories messages based on content

IV. Complex context events

For example, someone is approaching to an office.

3.2.2 USE CASE 2-ANNA

The second use case focus on young people with multiple, shifting interests, and flexible working hours.

- **General Description:**

Anna is a student. Her work time is flexible. Her activities in a day include doing research, as well as engaging her interests. It is assumed that lab room is a major sign of doing research. When she is doing research, less important messages should be pending.

- **Summary**

Context:

11:00 a.m. to 1:00 p.m. At laboratory.

1:00 a.m.-2:00 p.m. Home. Cooking.

2:00 p.m. – 3:00 p.m. Tennis square.

Interests:

Cooking, swimming. But after playing tennis, Anna begins liking tennis

Simulated Messages:

1) 11:00 Facebook messages about tennis events.

// deleted

2) 11:01 software update messages.

// pending

3) 11:03 text messages from close friends about catch-up.

// pending

4) 11:05 Facebook messages from close friends about catch-up.

// delivered.

5) 12:01 Facebook messages about cooking tips.

// pending

6) 1: 30 Facebook messages from friends.

// delivered

7) 2:03 email from her boss about catch-up meeting.

//delivered

8) Facebook message about tennis events.

//delivered.

- **Rules**

Complex rules are distilled from use case 2:

- Achieve precision between interests and message itself.
- Upgrade the level of interests.

3.3 REQUIREMENT ANALYSIS

The requirements of the system generated from user cases are listed here:

1) Different types of messages should be uplifted into a uniform and sharable format.
   The importance of different messages should be quantified. From the value, the system can give an operational decision to this message.
   In general, the order of the importance of messages from domain knowledge should be: text > email > facebook > software message.
2) The content and subject of messages influence the importance of the processing messages.
   In the system, it can be processed by importing the library of word list to help determine the meaning of content and subject. However, the library can be user-approved and system-defined. It has personalised, changeable library influenced by rules.

3) Categorised the messages

   It is necessary to categorise the message into different groups for further treatment. (i.e. John considers messages from family is more important than messages from friends in this case)

4) Related messages to a particular person

   Given that Mary's approaching to an office, it is necessary to pull all the messages related to Mary. It is to explore the relationship between ontology and rules.

5) Shifting interests

   About user scrutability, it is necessary for users to see the reasoning procedure and change the decisions.

6) Set the priority of rules.

   For a stable, high-performance system, it is necessary that the rules are executed in a reasonable order.

7) Large amount of messages

   In the real case, it can have a large volume of rules and messages, so it should have a simulator software to simulate an increasing number of messages in order to test the extensibility and scalability of the system.

   The system should have the capability to handle with a large amount of messages with acceptable performance.

8) The reasonable framework is required

   Given that Jess rule engine executes rules in a random order, it is recommended to build a system that can avoid iterate all the rules, which can reduce the cost and response time.

Chapter-4 **DESIGN**

The rule-based approach builds the foundation of message assistant system. It is a feasibility test system integrating personal ontologies and dynamic contextual ontologies. To do this, we need an effective architectural design to match ontological facts with rules effectively.

4.1 **INTRODUCTION**

After analysing the requirements of the system, this section will start with the abstract architecture design. Essentially, this system is based on rule-based approach, thus, the architecture is designed centred on improving the performances of matching rules and messages. Two architectures (centralised one and phrased-process, distributed one) are compared which have different weaknesses as well strengths.

Then, take the first use case as an example, detailed designs are given in the following part of this section. The designs are composed of two parts: one is designing ontological models, which contains models for raw-message uplift, models for matching dynamic user preference, as well as fast changing contextual models, aiming to interact with internal rule engines as well as external environment and user; the other is rule reasoning by giving examples and discussing the potential value of jess rule engine and backward chaining.

The final part will discuss some challenges in the current design. As a conclusion, some possible solutions and analysis to meet these challenges.

In summary, personalised, personalised, rule-based approach laid the function for the current application - message assistant system. Both centralised and distributed rule system has the strengths in executing rules which are illustrated in section 3.2.2. In section 3.2.3, a detailed design consideration for ontology representation is given. Many design challenges about ontology model are raised and solved. Section 3.2.4 focuses on backward chaining in Jess rules by giving several reasoning examples. In section 3.3, an overall analysis about the message assistant is described.

4.2 **ARCHITECTURE DESIGN**

In this part, two type of architectures to meet the requirements of message assistant system is illustrated and compared. One is centralised while the other is distributed chaining. Both of them has advantages and disadvantages.

4.2.1 GENERAL DESIGN

Underneath the high-level architecture, the process of handling messages is same listed as followed(see Figure 2): generate messages formed in metadata; message uplift (matching keyword list, remove redundancies and uplift persons); give the initial score of the message depending on the message ontology.(type, subject, sender, etc.);increase the score of the matched message depending on the user interests and context descriptions; category the message; increasing the score of the matched message, in according to incoming context event, changing personal preference trigger event. The system will calculate the score of every message in run-time.

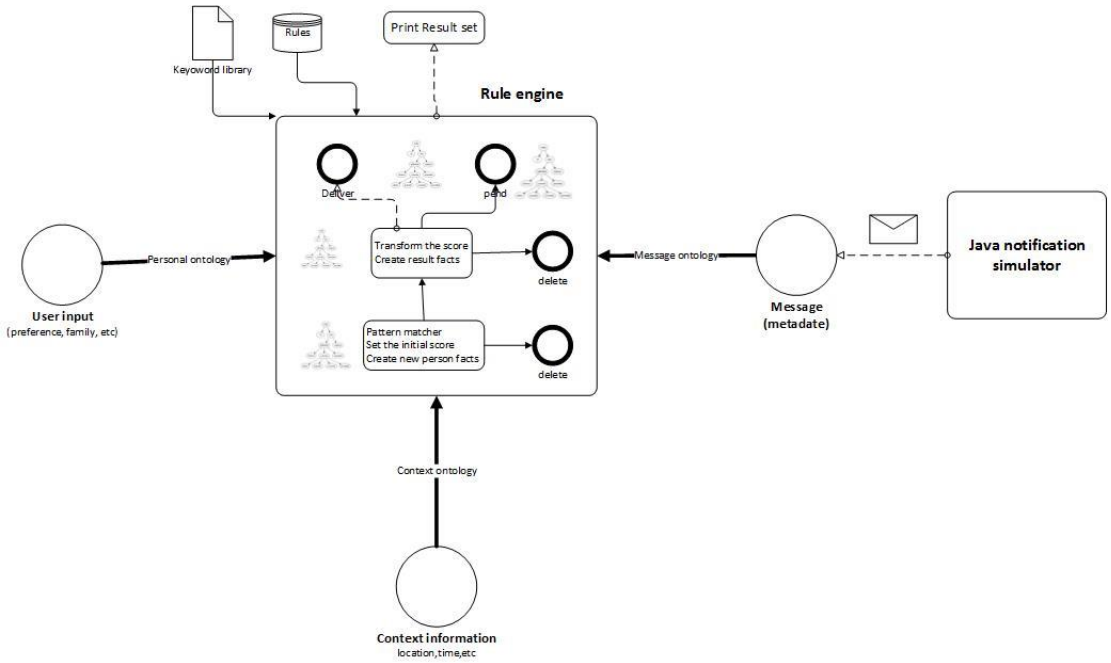


Figure 2: Message Handling Process

Conclusion: Three main ontologies are identified here; they are context, personal (user input), message ontologies and three of them support runtime fact transformation and fact assentation. After matching rules to facts in the working memory, the system will make corresponding operations on messages based on the importance (score) attributes in the messages. Importantly, the score is monotonic increased when the match rules are triggered. So if the score of messages is higher than delivered value during the routing procedure, the messages will be delivered; otherwise, the messages will be pended or deleted along the route.

The following part will compare the two designs from scalability, performance, load balance, fault-tolerance and security.

4.2.2 ARCHITECTURE 1-CENTRALISED DESIGN

4.2.2.1 Description

This architecture is centralised. All the rules and keywords library are imported into working memory to form into facts.

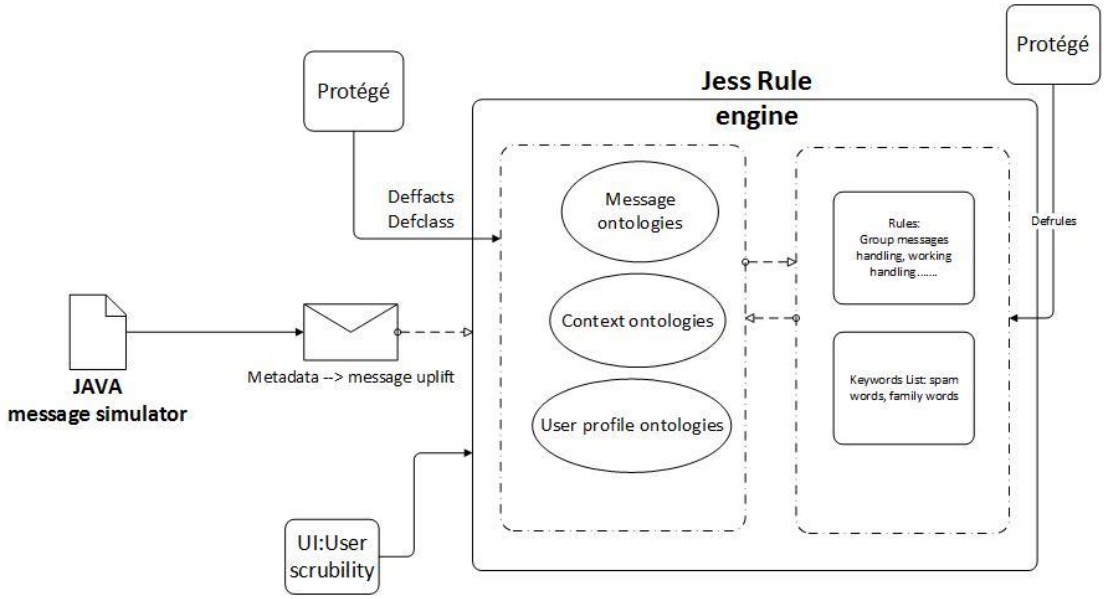


Figure 3: Architecture 1-Centralised Design. Knowledge and Ontologies Are Stored and Processed In the Working Memories

The process of the handling is (as Figure 3): Java simulator generates messages. These messages come to the main logic system where we use the development tool, protégé. Matched rules with message facts to increase scores. Then the system makes operation to message based on the scores with the messages in any stage.

To begin with, matched rules with message facts to do the initial operation. Afterwards, context facts and personal facts will be matched to increase the score of messages facts in working memories.

Five main and basic rules in the system are listed here:
(***Note***: In order to display the features of rule engines which can execute matched rules in any order, we list rules the order randomly)

- Categories A: Transforming the score of messages based on user interests:
  - Rule 1: If the message is from an important person, the score of the matched messages increases.
  - Rule 2: If the messages are about user interests, the score of the matched messages increases.
- Categories B: Transforming the score depending on context information:

- Rule 3: If the user is during working state and in the office, messages from work increase the scores.
- Rule 4: If someone approaching, all the messages related to the person increase the scores.
- Categories C: Transforming the score of the messages, depending on messages:
- Rule 5: Messages of different type influence the scores. Text > email, etc.
- Rules 6: If messages contain important string, the score will be increased.
- Rule 7: If some messages state similar content, the messages will be grouped together and the score will be added as well.

The execution of rules looks like Figure 4.

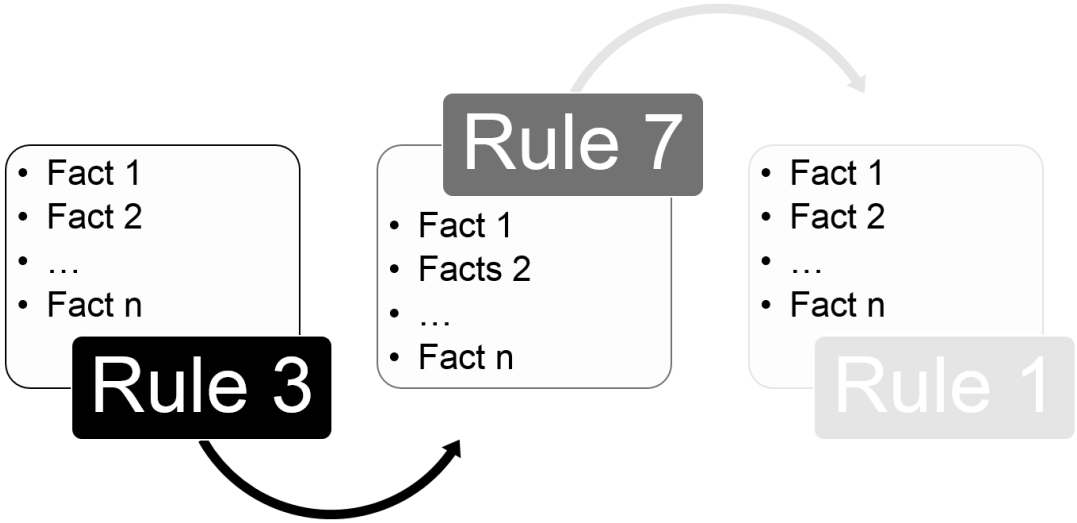


Figure 4: Execution of Rules. Facts are stored in the working memory

4.2.2.2 Analysis and Discussion

- **<u>Performance:</u>**

  Response time of the system is random for the given rule. But the average time is half of whole processing time of all the facts in the memory.

  Jess rule engine will decide what rules to fire and when. However, the pattern matcher will search the millions of rules which satisfy the rules, even though the rule contains only two or three premises. To make it more random, as to the priority of rules, different rule engines have different approaches to deciding which rule will be fired first.

  In this case, the optimistic result is that only one rule is fired. In the worst situation, all of the rules are fired, the messages are still in a pending state to wait for a trigger event.

- **Cost:**

  However, when rules are fired, all the facts in the memories will be matched repeatedly. The weakness is obvious that the cost of the system is high.

  One positive side of using rules is that rule-based systems are declarative programs which are light, requires some other runtime system to control flows.

- **Fault tolerance:**

  Because all the facts in working memories will be executed for a given rule, such all-effort will guarantee the correctness of executing the result of messages.

- **Security:**

  Just as every centralised system, the security is better than distributed one. However, the system needs much more complex security consideration.

  For example, the working memory store use-defined keyword list, as well as system default keyword list. Once the library information that user-defined is stolen by a bad-intended organization. It will cause many problems. A better way to resolve the issue is to authorize user's access and secure the fact base and rule base.

4.2.3 ARCHITECTURE 2- DISTRIBUTED DESIGN

4.2.3.1 Description

Distributed architecture will focus on avoiding the problem of randomness in centralised architecture, which helps to reduce expensive cost on the system. Besides, like all the distributed system, this architecture will bring other benefits, 1) loading the balance of the system, 2) extending to handle large numbers of messages and rules.

The architecture is displayed in Figure 5.

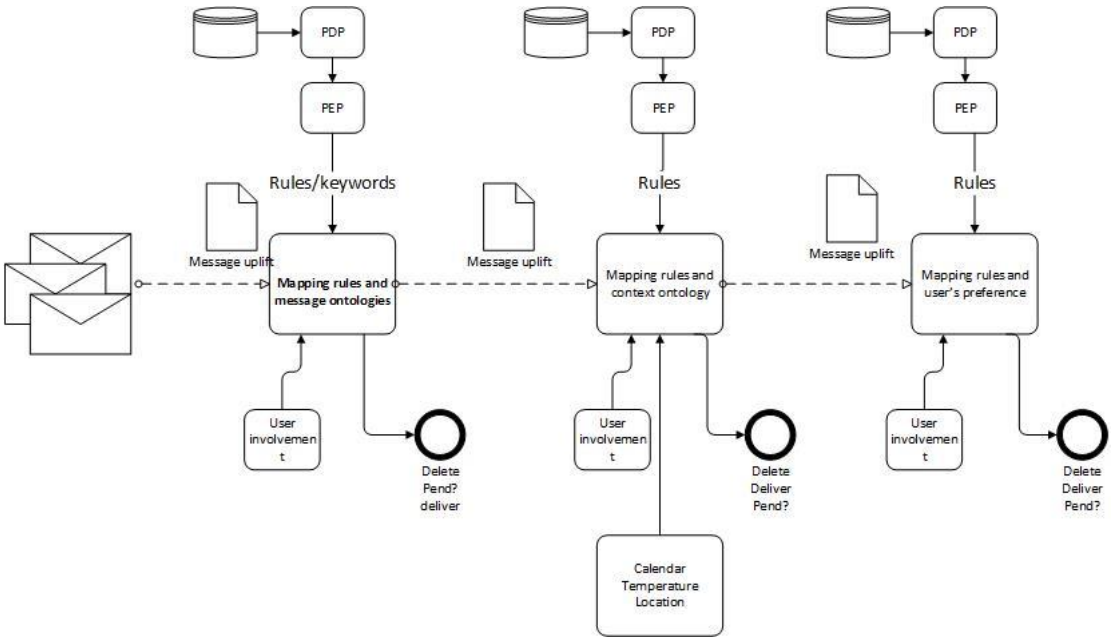


Figure 5: Distributed Design

The system is composed of three units. They are messages subsystem, context subsystem, and personal subsystem.

Every subsystem only contains related facts, rules and word libraries. Every subsystem can make operational decisions (delivered and deleted) upon the incoming message facts. It supports scrutiny of users and dynamic context.

General Description:

Several messages (metadata) first come into message subsystem, where matched the rules and person facts. After executed in the rule engine, messages are attached with phrased scores. If the score is high than the delivered value or lower than deleted value, the message will be removed from working memories, create a result set and be deleted/delivered. Once the score is between delete and deliver value, message subsystem cannot make a decision on this message. Then subsystem will flag the message and transfer the message facts to the context subsystem, and if the result is still unsure, then it will be transferred to the personal subsystem. Context subsystem and personal subsystem will execute similar operation like messages subsystem.

If some event is triggered (for instance, context changes), the message's score get increased in a particular subsystem (in this case, context subsystem can execute the final operation to the message facts). Then this subsystem will make an operation to the message and notify the score of the facts and the current state of the message fact to other subsystems. In the end, other subsystems will remove such message facts from working memories.

Several features of this system are listed：

In every subsystem, only related facts and rules are imported into the working memories. For instance, context subsystem has context facts and pending message facts. The benefits of such design is that not only it decrease the all-effort search time through the whole working memory, and response time in further step, but also reduce the cost of storing and matching all the facts in the expensive working memory.

Every subsystem can make an independent decision (delete or delivery) based on the current score of the message facts. It speeds up the response time and reduce pattern match in the following subsystems.

In response to intermediate event happened on pended message facts, the current working subsystem will notify the updated score and message state to all other subsystems. If the subsystem realised that the message has operated, the facts will be removed from working memory; otherwise, this message fact will be updated.

4.2.3.2 Analysis and Discussion

Essentially, it overcomes the problem of the centralised one. It categories facts and rules, which avoids complete search in working memory in the centralised architecture. However, if the message is in a pending state, waiting for a trigger event, the cost of the system is still unrealistic because it has to synchronise all subsystems and easy to make fault in the procedure.

- **Performance:**

  Just as mentioned in this section, two parts contribute to shorter response time than architecture 1. One is going through facts in subsystem rather than matching all facts in the whole system when a rule is fired; the other is subsystem can make operational decisions on current message facts, cutting down on time on pattern matcher.

  Another benefit in performance is decreased system cost. The explanation of this benefit is similar to that of reduced response time. A subsystem is responsible for part of facts matching.

  However, there are slight disadvantages compared with architecture 1. Facts of Pended message can be copied to subsystem more than once, increasing the memory cost and also lengthening the response time due to the time spent on transferring facts.

- **Load balance:**

The balanced load reflects on rule repository, keyword base, working memory. Due to this fact and rule distribution design, the system becomes able to take more tasks which increase system performance.

- **Scalability**

  More capable of handling large numbers of messages and rules than a centralised system. In addition, if new situation is required, such as dealing with video messages, a new subsystem focus on video can be added after the message subsystem, bring the benefits of extensibility and scalability.

- **Fault tolerance**

  The system has not given enough consideration to faults. But generally speaking, distributed design is easy to make more fault than the centralised one, because distributed system has high possibility to lose connection and lose packets.

- **Security:**

  The distributed design needs security consideration because keyword library and rule repository are easy to be exposed. In order to guarantee the security of user access, the system can adopt public key mechanism.

In summary, both centralised and decentralised architecture have advantages and disadvantages. From the point that the system needs to be scalable enough to handle large numbers of messages, architecture 2- distributed one is ideal than the centralised one.

4.3 ONTOLOGICAL MODELLING

To seek the common understanding of different types of information, the semantic ontological models should be shared by many applications. In addition, the design of ontological model should consider the complete, self-reasoning capability. In this section, four main ontologies are identified. In the design of each ontology, key challenges are identified. In the last part of the section, ontology and OWL to reason personal relationship in order to compare the reasoning ability, information completeness of the two ontology languages.

The overall ontologies are formed into the following graph:

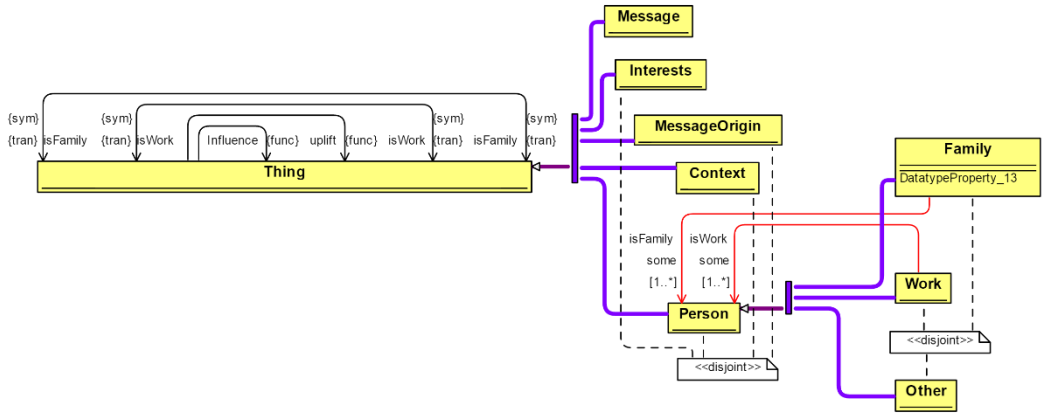


Figure 6: The Ontology Graph

4.3.1 MESSAGE ONTOLOGY

The requirement of this ontology is to uplift information which helps to make an operational decision, set score attribute to determine the decision.

The ontology should be reusable, which receives fast changing messages quickly.

Two main ontologies related message handling are designed. They are MessageOrigin.class which generates facts from metadata messages, and Message which uplifts key information and person from MessageOrigin.class as well as set score attributes onto facts.

a) Challenge 1: The way to uplift key information is to match contents of message with a keyword list. However, some important keywords may not pre-exist in the keyword list.

Solution: user can check the reasoning process and assert user-defined keyword into the working memory;

b) Challenge 2: Person may appear in the system at the first time, it is necessary to assert new person facts into the system.

Solution: use rules to add a new person into the working memory as well as initialise some properties and attributes of the person.

4.3.2 PERSON ONTOLOGY

The requirement of person ontology is to record personal profile information, personal relationship including work people list, family people list.

Key challenge: Referred to user case 1, John put more propriety to messages from family, so it is necessary for the system to reason whether the sender is from family or work.

Solution: in protégé ontology, it can use multi-slot to put a person from the family into family slot list. However, such object-based way has its limitation to express the relationship between people. Another solution is using OWL, this language supports rich reasoning syntax to make ontological reasoning effectively.

Compare protégé ontology and OWL in exploring the personal relationship.

In protégé ontology, we can assign person related to the user into a correct group, using multi-slot.

However, in OWL, the family relationship can be reasoned from SWRL rules, combining properties and class. The detailed design and implementation of OWL is illustrated in section 5.3.4

Note: The person ontology records sender's information and relationship with the user. So personal ontology does not contain information about user information.

4.3.3 INTEREST ONTOLOGY

This ontology aims to quantify user's preference. So it has an importance property to mark the importance of the state interest.

In addition, interest ontology should be reused, and able to deal with dynamic user interests.

4.3.4 CONTEXT ONTOLOGY

Context includes current state information, such as location, time, etc. The dynamic contextual ontology represents series of events.

The requirements of context ontology are to represent a various type of context information, link event participants to corresponding person ontology.

The key challenge is how to connect event participant to person ontology.

Solution: it will leverage rules to link current facts to other facts in working memory.

In addition to these ontologies, the working memory is also imported Keyword library to help uplift important String and set scores to messages.

4.4 RULE

The rule-based system adapts forward chaining. In forward chaining, all the rules which can fire do fire. Forward chaining can lead to breadth-first search which wastes lots of costs and increase the response time. The benefit is that user can see the process of rule handling which is necessary for message assistant to scrutinise the routing path of rules.

Even though the system uses forward chaining, the design analysis can use the approach of backward chaining.

$$\left.\begin{array}{l} Keyword\_match \Rightarrow Score\_add \\ \wedge Context\_match \Rightarrow score\_add \\ \wedge Meet\_Interest \Rightarrow score\_add \end{array}\right\} \mapsto Score > delivered\_value \mapsto deliver! \quad (1)$$

So In order to meet the high-level objective of operating messages, we need message uplift and keyword match and calculate the score of messages. Finally, the system operates the messages based on the score.

Several rules are identified as followed. Among them, three of them are challenging than others, giving detailed design explanations.

Rule 1: add new person

Description. If the sender did not exist in the working memory, the sender information will be added.

R 1: If $Sender \notin Person \Rightarrow$ make- instance of Person (sender). (2)

Rule 2: message uplift

Description. After receiving raw messages, match the sender with person facts, and match contents with message facts.

$$\text{R 2: If, } \left.\begin{array}{l} sender \in person \\ Content \in Keyword\_list \end{array}\right\} \Rightarrow \text{ make-instance of message} \quad (3)$$

Rule 3: give initial score of message facts

Description: Based on content, type and urgent, give the phrase score of the messages.

$$\text{R3: If } \left.\begin{array}{l} content \in work/\ family \vee content \in spam/normal \\ type \in\ facebook/text/email/app \\ urgent == true \end{array}\right\} \Rightarrow \text{ slot-set: importance} \quad (4)$$

Rule 4: category the messages.

Description: category the message into family and work group, and set the score of the messages.

$$\text{R4: If } \left.\begin{array}{l} sender \in family\_list \\ \wedge\ personalRelationship \neq nil \end{array}\right\} \Rightarrow \begin{cases} slot-set: personalRelationship \\ slot-set: score \end{cases} \quad (5)$$

Rule 5: content event influence the score of message facts.

Description: messages imply work (from the time, location) or resting will increase the score of message facts. Work and resting will increase the score by different values.

$$\text{R5: if } \left.\begin{array}{l} location \in work/rest \\ \wedge\ time \subseteq work/rest \\ \wedge\ other\_evidence \end{array}\right\} \Rightarrow \text{ slot-set: importance} \quad (6)$$

Rule 6: Context- someone approaching user

Description: When someone is approaching, the messages related to the person will be delivered.

$$\text{R6: If } \left.\begin{array}{l} approaching\_person \in Person \\ \wedge \sum Message(approaching\_person) \end{array}\right\} \Rightarrow \text{slot-set: importance} \quad (7)$$

Rule 7: Upgrade similar messages

Description: a message of similar content will upgrade the score of message facts and remove duplicates.

$$\text{R7: If } \left.\begin{array}{l} m1 \leftarrow message(content) \\ \equiv m2 \leftarrow message(content) \end{array}\right\} \Rightarrow \begin{cases} slot-set: score \\ remove-instance: m1 \end{cases} \quad (8)$$

Rule 8: upgrade messages

Description: If the message contains keywords about user's current interests, the importance of the message will be upgraded.

$$\text{R8: If } \left.\begin{array}{l} Content \in Interest \\ \wedge\ Interest \in Interests \end{array}\right\} \Rightarrow \text{slot-set: importance} \quad (9)$$

The figure illustrates the detailed implementation of rules handling interests. The steps are identifying interests, and upgrade the importance of messages.

Rule 9: Transform the standard of importance

Description: In order to let system adapt to user's current interest and habit, the value of weight standard should transform.

R9: Transform weighing values.

Chapter-5 IMPLEMENTATION

5.1 INTRODUCTION

The previous section gives the detailed design of the message assistant system using personalised, rule-based approach in handling overflowed messages. The design is derived from the state of the art, comparing and discussing the novel and solution to challenges in current research and applications. The research questions and key challenges raised in section 1 will be solved in this section. This section will implement the message assistant system in an iterative development way, concentrating on the effectiveness of rules and information representation. The following sections will evaluate the approach by conducting several experiments. In the last section of the thesis, some discussions on the current work as well as future work are illustrated.

The major research goal is to explore the feasibility of rules in handling messages, taking account of user preference and context. In response to the requirement, evolving rules developed iteratively are tested with test cases and proved that it is feasible and efficient to adapt rule-based approach.

The following section will be divided into two implementation scenarios, corresponding to two user cases. Different focuses are put on the two cases. The first user case centred on rules processing facts from the same class, for example, accumulating similar messages. It also focuses on interrelated facts of differing classes, for instance, a complex context which a person approaching users will trigger all the message sent from the person. However, in the second user case, rules take an effort to handle dynamic facts, for example, users is able to inspect and change the priority of interests. The order of fired rules will be kept watching as well as the number and order of influenced facts.

A set of experiments are conducted to test the rule-based approach centred on two user cases, and evaluations when extending to large numbers of messages. In addition, there is an extra experiment that writing a simple version implemented using OWL based on user case 1, intended to compare effectiveness and efficiency between OWL and protégé instance.
The experiments timeline is given in Table 1.

Table 1: Experimentation and Implementation

	Description	Message simulator	Message generator	Ontology model	Rule engine
Stage 1-1	Implementation - Use case 1	None	None	Implemented	Implemented
- Exp. A	Rule 1,2,3	None	None		
- Exp. B	Rule 4,5,6				
- Exp. C	Rule 7				
Stage 1-2	Implementation - Use case 2	None.	None	Implemented	Implemented
- Exp. A,B,C	Dynamic Interests. Rule 8,9	None	None		
Stage 2 Large numbers	Further implementation- Use case 1	Implemented	None	Improved	Improved
- Exp. A	Rule 5	Implemented	None		
- Exp. B	Rule 4	Implemented	None		
- Exp. C	All rules	Implemented	None		
Stage 3- Novel messages	Further implementation –Use case 1	None	Implemented	Improved	Improved

The next section will begin with message simulator, a Java application. The major requirement of the simulator is to generate conditional messages after filling in the total number of messages, time period, the urgent number as well as the number from family.

Part 5.3 will give a detailed implementation of user case 1, and general description on experiment A. In this section, some challenges and the level of completeness are listed. The

process and architecture are given at first. Followed by employed technology description, implementations of information representation and uplift using ontological models are detailed illustrated. This section will end with implementing evolving rules.

Part 5.4 followed the similar pattern of part 5.3. However, the rules and ontological models are different from those in part 5.3. The detailed implementation will be given in that part.

5.2 MESSAGE SIMULATOR

In order to explore the effectiveness of the message assistant system, a message simulator which generates various messages is required.

5.2.1 REQUIREMENT

The high-level goal of the simulator is to generate random messages from messages or apps in metadata form.

The detailed goal of the simulator is that the simulator can generate different types of messages in a different time period, etc. It is used for testing the effectiveness of particular rules.

The requirements are derived from the goal.

R1: To conduct evaluations, these messages should have conditions to limit types and the number.

The parameters of the simulator are listed as followed:

- The number of messages
- The urgent number of messages
- The number of messages from family
- The time period of sent time

R2: In order to test the fault tolerant of the message assistant system, random novel messages should be generated.

R3: support messages from family.

In order to support selecting messages from family, given that the simulator cannot figure out whether senders are from family or not, the system can establish data bases for family messages.

5.2.2 DESIGN

Based on the requirement analysis, the simulator has three main components:
Message base (normal, family, novel), Graphic user interface (GUI), generating engine, output to protégé.

Components:

The message base stores the message in general form. The storage can be databases, XML, etc. Graphic user interface (GUI) is used for providing an interface for testers to fill in filter conditions for messages.

Generating engine is responsible for reading messages from message base, receiving parameters from GUI components, and output messages in standard form.

Output to protégé supports transferring messages to rule development environment, protégé.

Timeline:

The process of the java application: After receiving the parameters, generating engine will connect the base to execute filter algorithms to obtain messages matching the conditions. Then the engine will transfer the format of messages and send to Protégé.

5.2.3 IMPLEMENTATION

From the class descriptions, the important functions of these class are listed as followed:
The GUI receives the parameters and the graphic is display in Figure 7.

Figure 7: The User Interface of the Message Simulator. A sample input is given

The running result is in Figure 8 and 9. As shown, XX new messages are exported to protégé.

So the test environment is partly built.

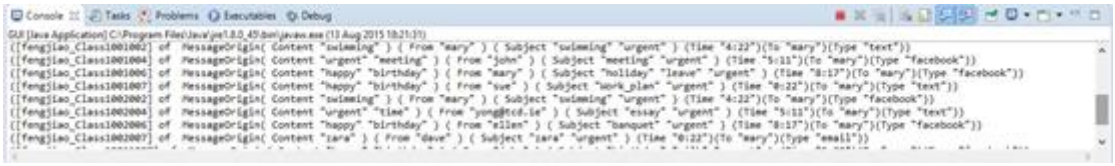


Figure 8: Console-Running Result of the Message Simulator.

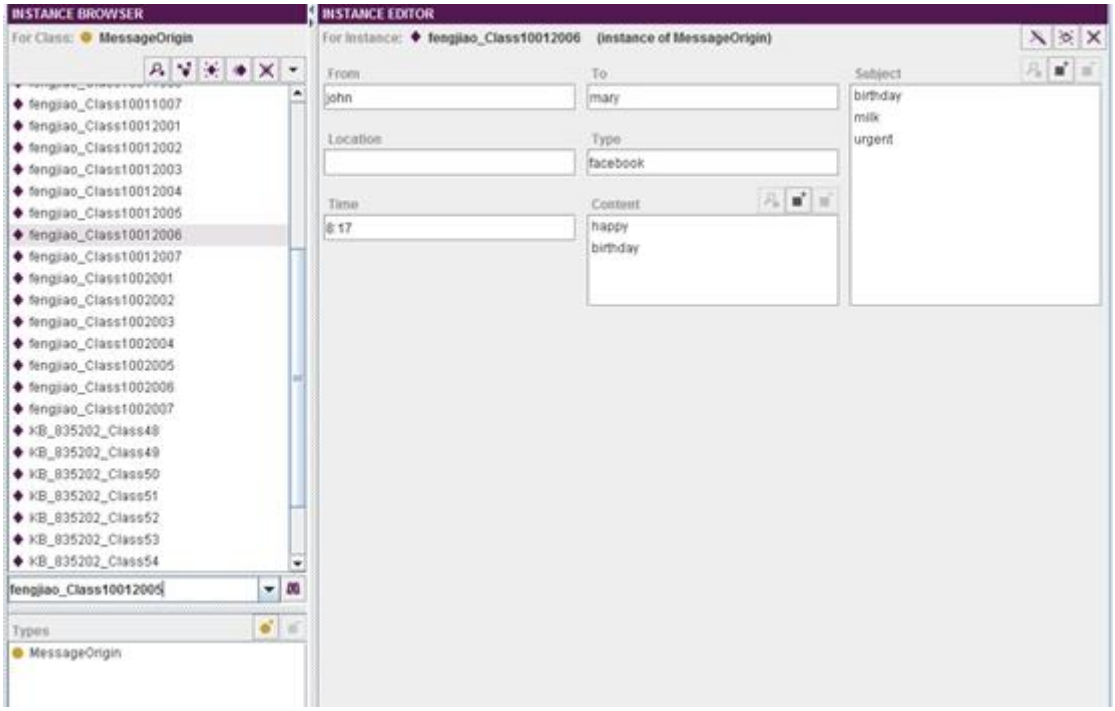


Figure 9: Protégé-Running Result. Fengjao_Class10012005 is A New Message Instance Generated by Message Simulator

5.3 IMPLEMENTATION OF USE CASE 1-JOHN

This part will give a detailed implementation of user case 1, and general description on experiment A. In this section, some challenges and the level of completeness are listed. The process and architecture are given at first. Followed by employed technology description, implementations of information representation and uplift using ontological models are detailed illustrated. This section will end with implementing evolving rules.

5.3.1 CHALLENGES REVIEW OF USE CASE 1

The first user case concentrated on white-collar people, who need the system to help them priority incoming messages. People worked in the office has the need to keep focus.

The implementation of this use case followed the research objectives and design requirements. The objectives of this case is listed as followed:

- Implement a high-level system to make an operational decision (including delete, deliver, or pending until conditions are satisfied) on incoming messages.

- Implement ontology models for context, person, and message information as well as knowledge.
- Implement a rule-based approach for combining working facts and rules
- Implement an architecture to test the performance when there are large numbers of messages.

According to goals list above, the challenges for Use case 1 are summarised in the following:

Summary of the challenges for use case 1

	Descriptions of challenges:	Completion:
1	Complete ontology models	Partly completed
2	Reason personal relationship	Completed.
3	Handling group messages	Completed
4	Handling novel messages	Partly completed.
5	Category messages	Completed.

Integrating user and context ontology, the facts of messages can be filtered and operated in a personal way. Whether ontology models are complete is the first major challenges for any effective information system. The depth of the ontology tree and data type of every slot are related to the completeness of ontology models. In use case, John is only interested in whether messages from work, or from family. It is a simple requirement by assigning sender into slot lists. In some complex examples in the real world, this level of completeness is not enough. For example, John is interested in from co-workers, but a message from co-workers' co-worker may raise his interest as well. Because this message can be about work in high possibility. The level of completeness depends on the complexity of use case. In this message assistant, the ontology model is able to cover all necessary information and information relationship.

The second major challenge is the ability to handling novel information, and similar information. These metadata from messages are uncertain. It is necessary to category message based on content as well as handling novel ones.

According to the goals and challenges of message assistant, the implementation in Experiment-Stage-1 aims to investigate the rule-based approach stated in design chapter. In the experiment, the process includes messages import, messages uplift, personal relationship identification, message category, context and interests match, and message operation. The

novel message handling as well as dynamic context handling will be stated in further experiment of Experiment-Stage-2 & 3.

5.3.2 TECHNOLOGIES EMPLOYED

This section will give an overview of employed technologies as well as corresponding analysis.

Protégé

Protégé is a development environment for ontology modelling, developed by Stanford University. Protégé has many open plugins to support OWL, Jess, Visualisation, etc.

In particular, OWL plugin is used to edit OWL ontologies and SWEL rules

JessTab is leveraged to edit and process Jess rules.

Ontoviz, graphicviz, etc. are used to display graphic ontologies.

Protégé ontology: Class, instance

A basic protégé ontology contains class and subclass to describe objects. Inside of the class, slots/multi-slots are created to set attributes of the object. Instances are data in a knowledge base. In general, instances are created/entered after the class and slot structure has been built.

Jess and Jess rules

Jess is short for Java Expert System Shell, a rule system written in Java. Jess is an efficient, lightweight rule engine to process rules and solve many-many matching problems with Rete algorithm. In addition, Jess script is also compatible with CLIPS.

JessTab is also a plugin for Protégé to link Jess engine to Protégé. Particularly, it increases functions that enable mapping protégé information bases to Jess facts, instances. In addition, it supports manage protégé classes and instances.

5.3.3 IMPLEMENTATION OF ONTOLOGICAL MODELLING

As the design chapter illustrated, three main types of ontologies are implemented and described in detail.

The overview of these ontologies is displayed in Figure 10.

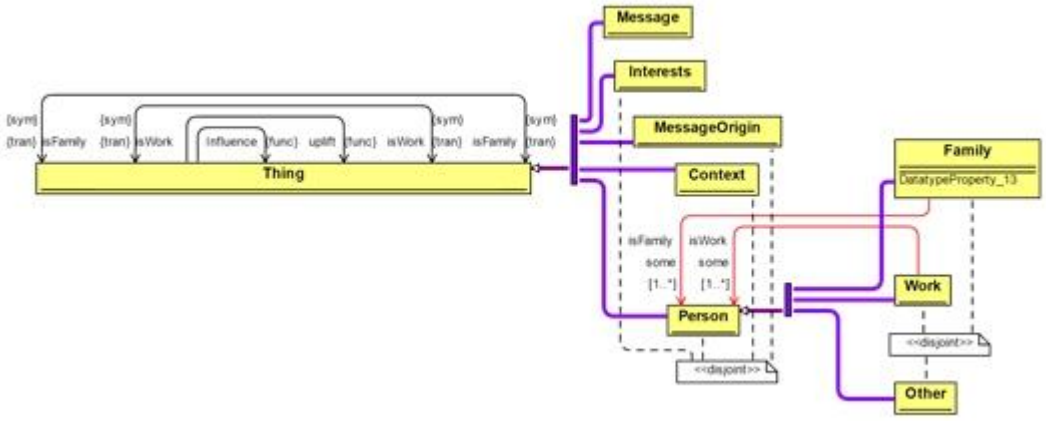


Figure 10: The Structure of Ontologies, Including Inheritance

All the ontologies are inherited from Class: Thing.

MessageOrgin.class describing original messages will be uplifted to Message.class, and the sender string will be matched to Person.class.

In Person.class, three subclasses are used to divide people related to the use into three categories. They are Family.class, Work.class and Other/Unknow.class.

In order to focus on the implementation process and key features of implementation, we will take MessageOrigin.class as an example to illustrate the detailed codes. The detailed codes about other ontologies will not be mentioned.

```
(defclass MessageOrigin
    (is-a USER)
    (role concrete)
    (single-slot Time
        (type STRING)
;+      (cardinality 0 1)
        (create-accessor read-write))
    (multislot Subject
        (type STRING)
        (create-accessor read-write))
    (single-slot To
        (type STRING)
;+      (cardinality 0 1)
        (create-accessor read-write))
    (single-slot From
        (type STRING)
;+      (cardinality 0 1)
        (create-accessor read-write))
    (single-slot Location
        (type STRING)
;+      (cardinality 0 1)
        (create-accessor read-write))
    (single-slot Type
        (type STRING)
;+      (cardinality 0 1)
        (create-accessor read-write))
    (multislot Content
        (type STRING)
        (create-accessor read-write)))
```

Figure 11: Implementation Details of MessageOrigin.class

Figure 12: Graphic View of MessageOrigin.class, Including Its Attributes and Relations

The ontology view is displayed in Figure 12. The subject and content are multi-slot lists, containing many string lists from original messages. Worth noting is that if the message is urgent, String "urgent" will be filled into the content list. However, it can bring other issues that senders state that the message is urgent, but the message does not belong to urgent one.

1) Message

Message.class is uplifted from MessageOrigin.class by matching keywords, which reduce the size of the incoming messages.

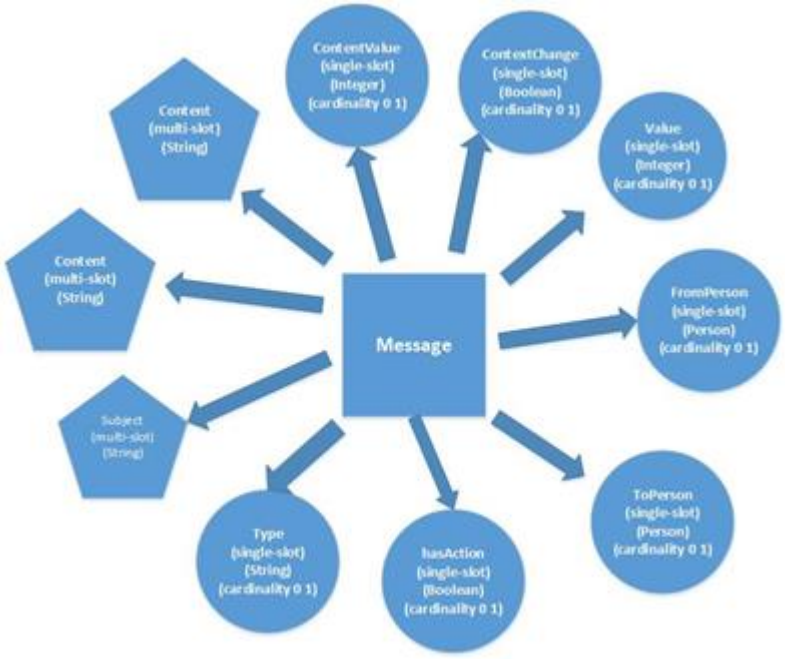


Figure 13: Graphic View of Message.class, Including Its Attributes and Relations

Before and after

Before

```
61  ([fengjiao_Class1001003] of  MessageOrigin
62
63      (Content "hiking")
64      (From "sue")
65      (Subject "hiking")
66      (Time "5:21")
67      (To "mary")
68      (Type "facebook"))
```

Figure 14: A Message before Uplifted

After

```
535  ([newMessage_Class20007] of  Message
536
537      (Content "hiking")
538      (ContentValue 0)
539      (ContextChange FALSE)
540      (FromPerson [sue])
541      (hasAction FALSE)
542      (PersonRelation "work")
543      (Subject "hiking")
544      (ToPerson [mary])
545      (Type "facebook")
546      (Value 40))
```

Figure 15: A Message after Being Uplifted

There are some changes from MessageOrigin.class to Message.class. Strings of content and subject are transferred and uplifted from MessageOrigin.class to Message. The context default information about the messages is imported into the message in order to track the context change. "HasAction" is used to detect another context event, when a person approaches the user. Importance of the message here is vital to decide in runtime whether the message can be delivered or deleted. In addition, sender and receiver information also transformed from string to person type after executing rule: message-uplift which match a string to an instance of person.

Besides, if the working memory fails to match an instance of person to the string of sender, another rule: new Person will be executed to create a new instance.

2) Person and Interests

Person.class is used to illustrate sender's information and sender's relationship with the user. However, Interests.class stores user's interests and preference of the interest.

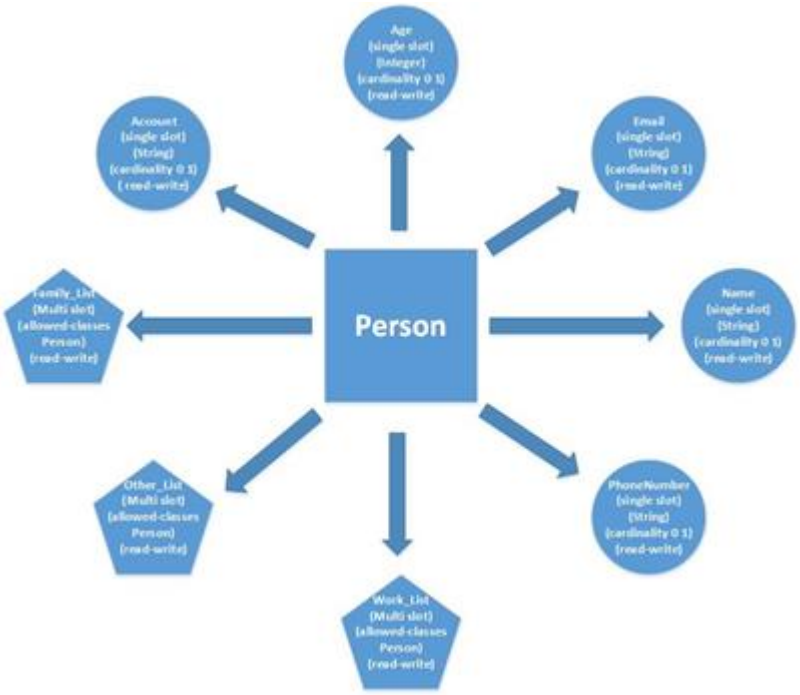


Figure 16: Graphic View of Person.class, Including Its Attributes and Relations

Person.class has slots to describe the basic information about the person and relationship information with other person stored in multi-slot lists (for example, Family_List). In the following section, there will be rules to match relationship in two related person facts. For instance, if A is in the Family_List of B, then B is in the Family_List of A.

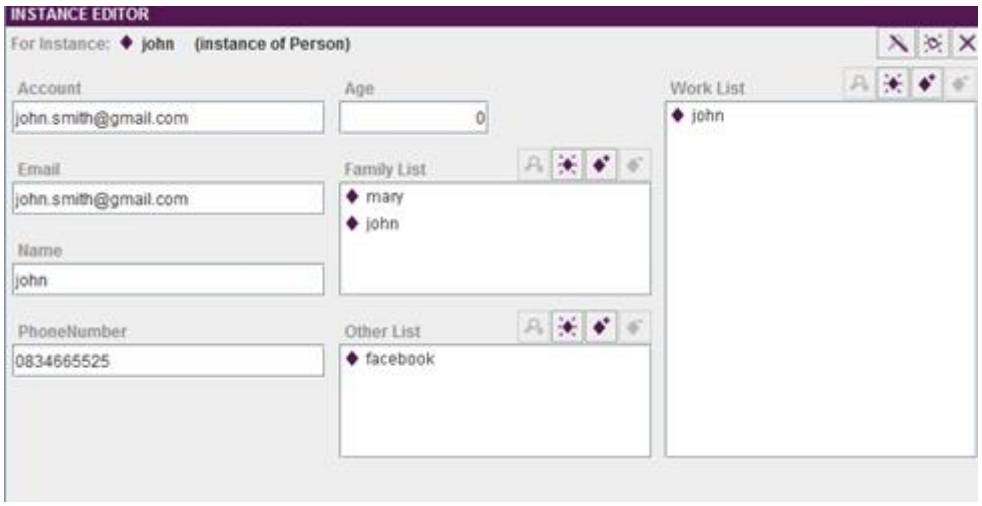


Figure 17: Protégé-John (An Instance of Person). Mary Is In The Family List of John.

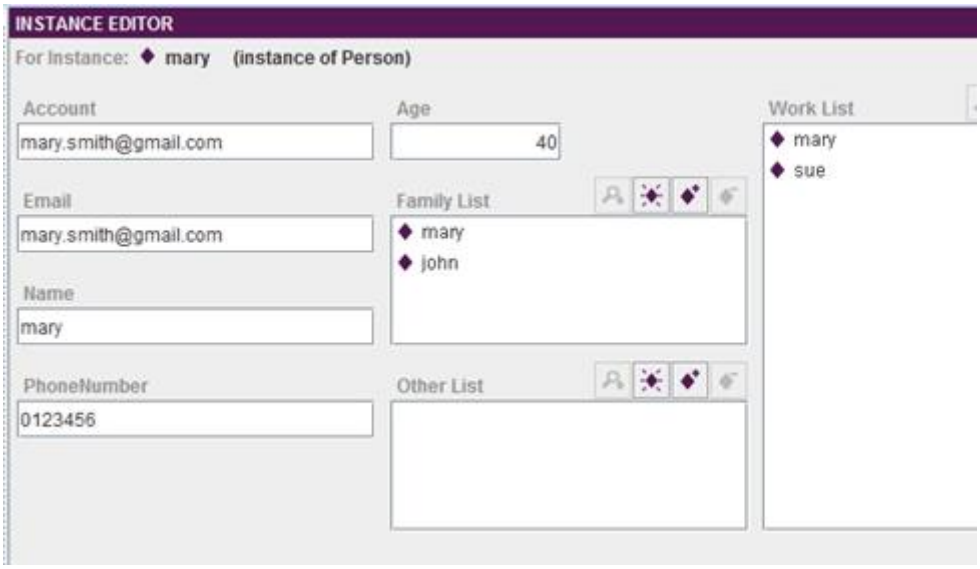


Figure 18: Protégé-John (An Instance of Person). Symmetrically, John Is In The Family List Of Mary.

3) Static base

The static base contains two types of static knowledge. First is Library.class which contains keywords list to category content and context (divided into three categories: Free_Location, Work_Location, Words_List). The other is weight. Class which contains value list about how to increase the importance of message.

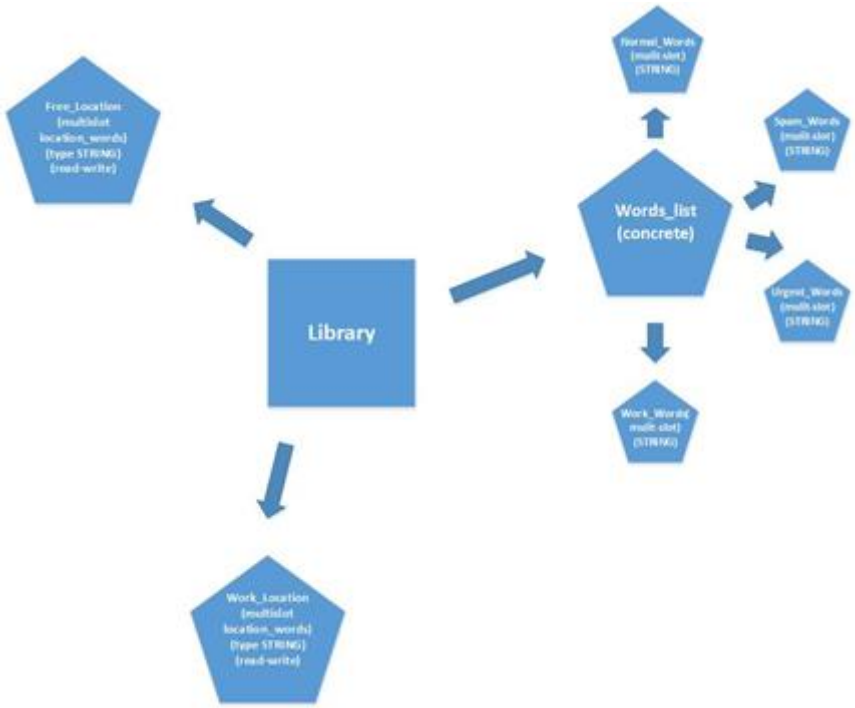


Figure 19: Graphic View of Library. Class, Including Its Attributes and Relations

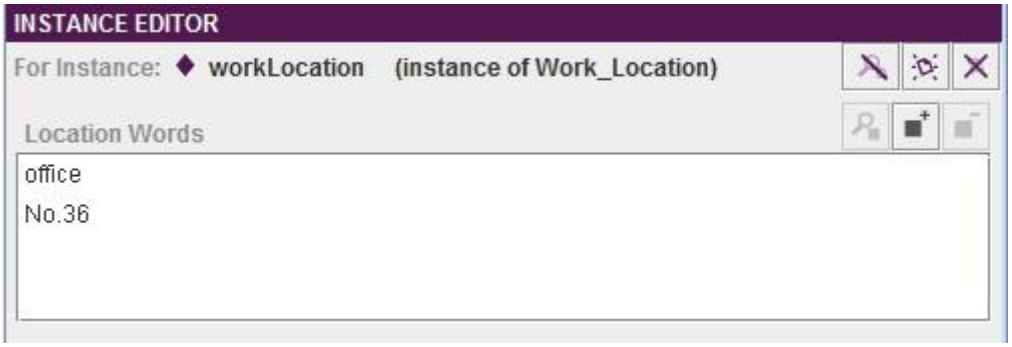


Figure 20: A Sample Example of the Instance of Work_Location

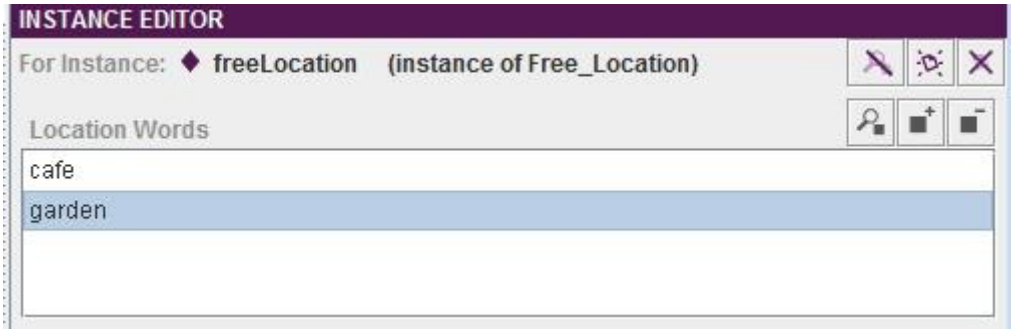


Figure 21: A Sample Example of the Instance of Free_Location

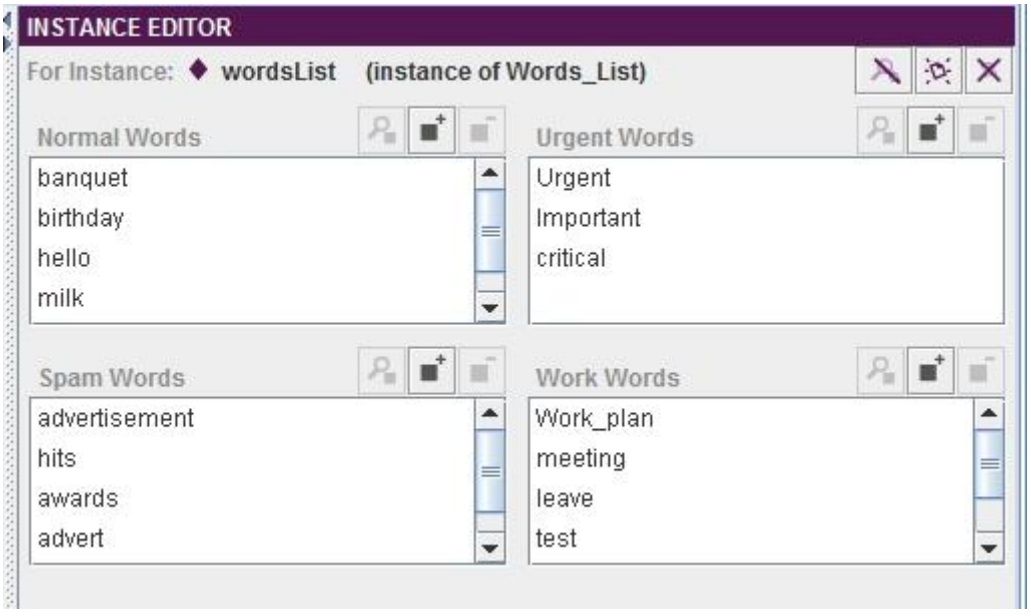


Figure 22: A Sample Example of the Instance of Words_List

Figure 23: Importance Standard- weightInstance (An Instance of Weight.class)

5.3.4 IMPLEMENTATION OF RULE ENGINE

Circled on key challenges, this part will focus on generating scores during rule developments. Rule approaches follow an iterative development.

Some challenged requirements and solutions of rules are listed in the table.

Table 2: Some Requirements and Solutions in Implementing Rules

Description	Solution
The system should be able to handle situations when there are novel people and novel messages.	When detecting new people, information about this person will be added into working memories.
The operational decision should partly depend on the importance of messages. The system should identify the importance of the incoming messages.	When there are incoming messages, the initial importance of the message will be calculated based on content and type of these messages.
To meet the requirement that messages from family should have higher priority, the system should group messages to know if they are from family or other categories.	Categorise messages into family and work group. The categorisation will base on two pieces of information: sender if it belongs to family list and personal relationship.
The operational decision is also related to current context. The system should	Context can be added into working memories. One particular context is that

dynamically asset context into working memory and increase the importance of messages if these conditions of context matches some messages.	when a person is approaching, messages related to the person will be delivered. It poses another challenge that the context instance should match person instance in run-time.
To meet the requirements of use case 1, another complexity is how to upgrade similar messages. It is required that similar messages should be combined, upgraded, and removed duplicates from working memory.	When a messages comes, the system will detect the working memory if there are some messages containing similar words. Then the incoming messages will increase its importance, and other pending and similar messages will be removed from the working memory.

5.3.4.1 Key features1- Adapt to novel messages

- Rule 1: Add new person

Description:

If the name of the sender does not exist in the working memory, an instance of the sender will be created and added into working memory.

R 1: If $sender \notin person \Rightarrow$ make- instance of Person (sender).

The implementation of R1 is displayed in Fig 24.

Figure 24: Detailed Implementation of Rule 1 - Asserting Novel Person

Executing result:

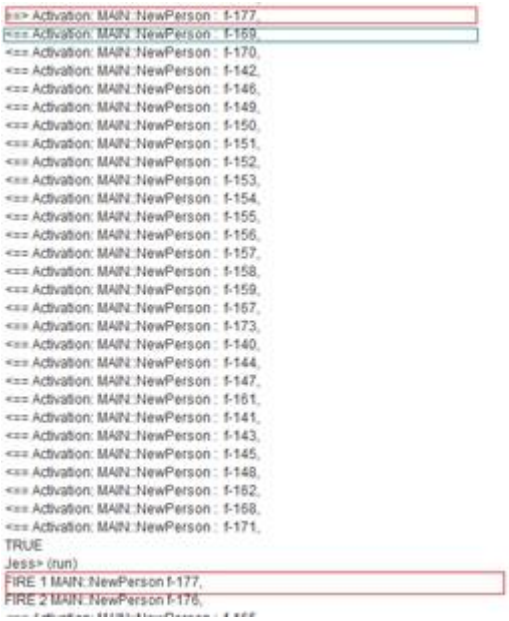


Figure 25: Compilation and Execution: Fact 177 Is Activated During Compiling Because Of No Such Person In Working Memory.

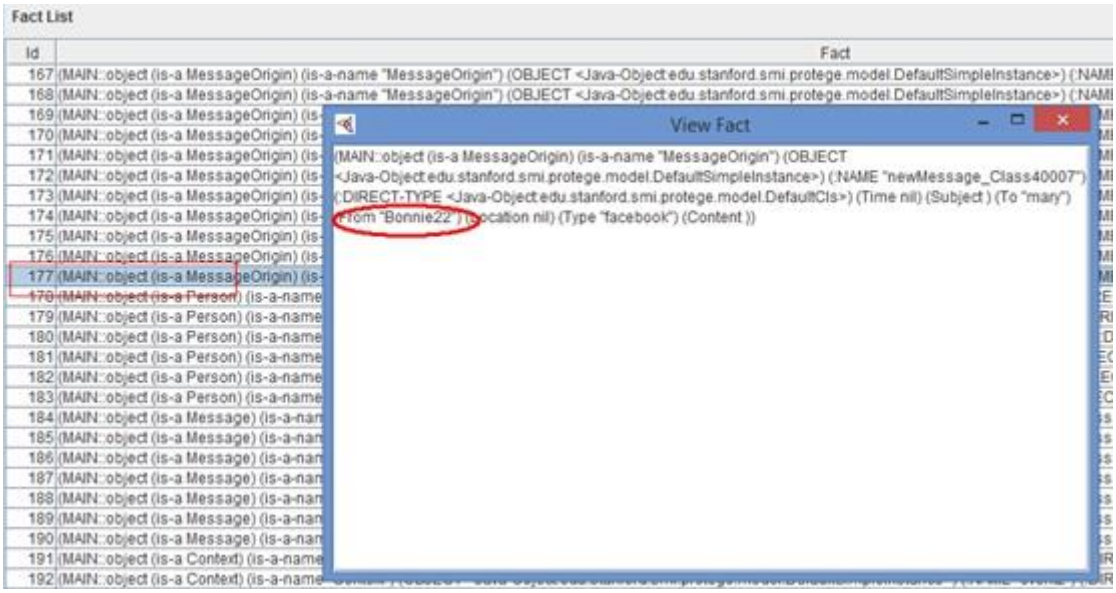


Figure 26: The Content of Fact 177 in Working Memory, Highlighting New Person Called "Bonnie22"

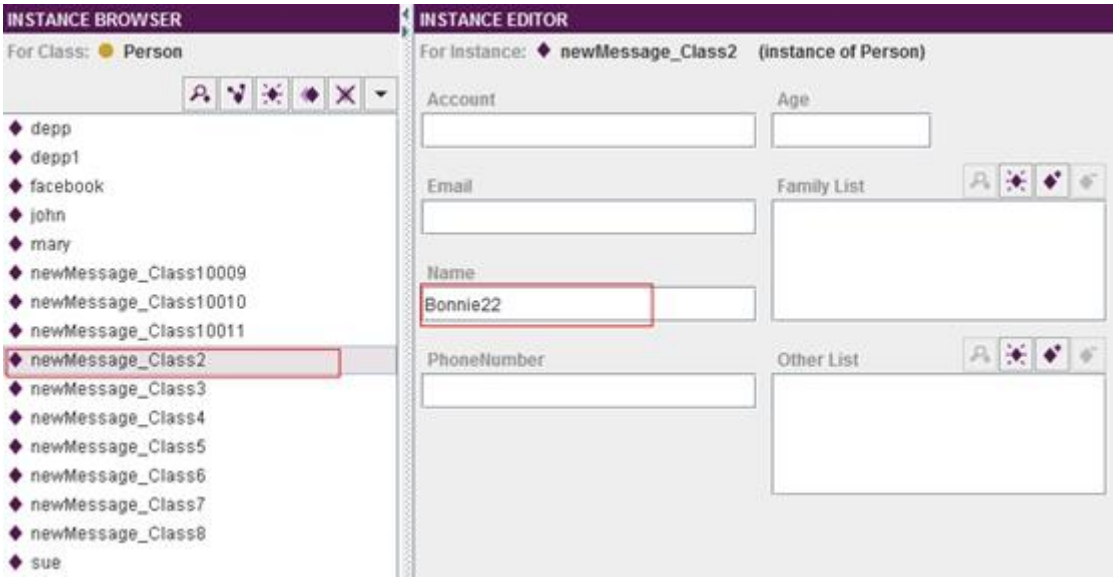


Figure 27: The Final Executing Result- New Instance of Person.class

Explanation:

As shown in Figure 25, f-177 is activated and fired because it contains a name string (bonnie22) that has not existed before. The rule 1 is fired and system inserts a new instance of Person.class into the working memory as Figure 27.

- Rule 2: Message uplift

Description:

After receiving raw messages, it will match the sender with person facts, and match contents with message facts.

R 2: If $\left.\begin{array}{l} sender \in person \\ Content \in Keyword\_list \end{array}\right\} \Rightarrow$, make-instance of message

The implementation of R1 is displayed in Figure 28.

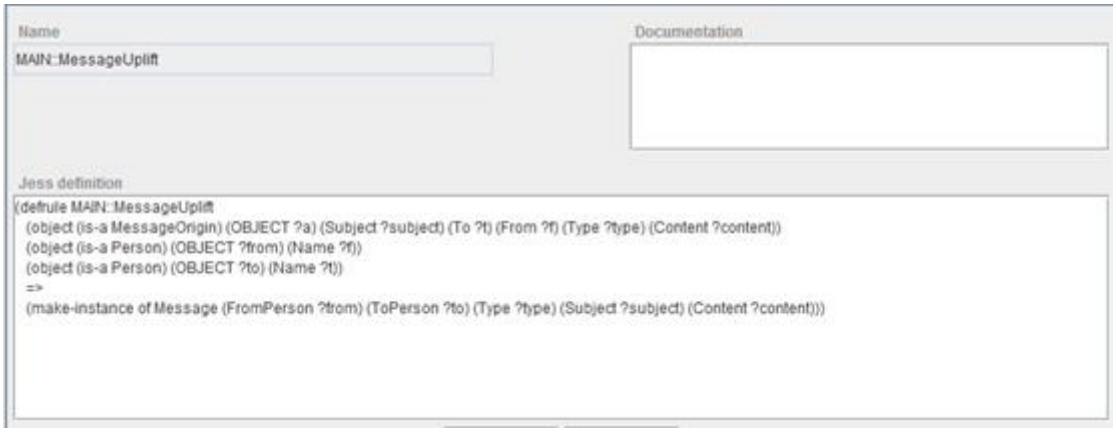


Figure 28: Detailed Implementation of Rule 2 - Uplifting Messages

- Rule 3: Give initial score of message facts

Description:

Based on content, type and urgent, it will give the phrase importance (score) of the messages.

$$\text{R3: If } \left.\begin{array}{l} content \in work/\ family \vee content \in spam/normal \\ type \in facebook/text/email/app \\ urgent == true \end{array}\right\} \Rightarrow \text{slot-set: importance}$$

The implementation of R1 is displayed in Figure 29.

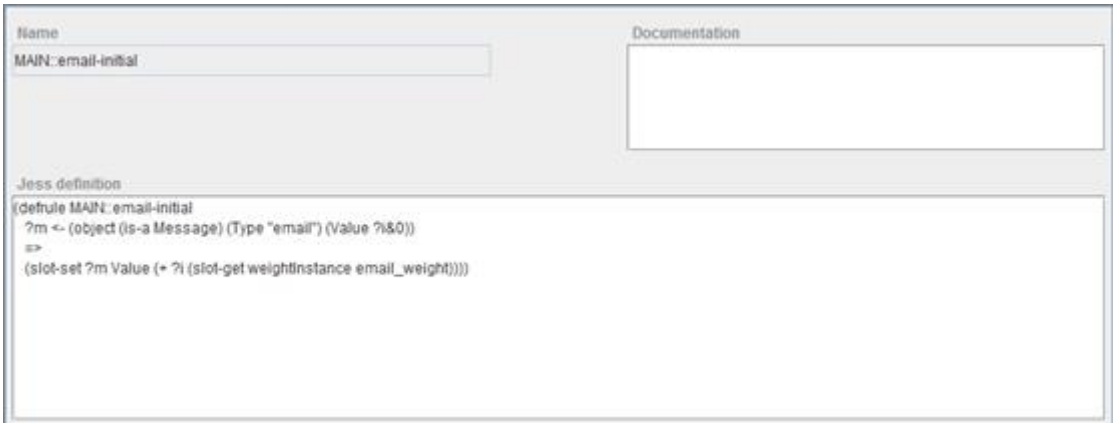


Figure 29: Detailed Implementation of Rule 3 - Initialising Messages by Type

- Rule 4: Categorise the messages.

Description:

Categorise the message into family and work group, and set the importance (score) of the messages.

$$\text{R4: If } \left.\begin{array}{l} sender \in family\_list \\ \wedge\ personalRelationship \neq nil \end{array}\right\} \Rightarrow \begin{cases} slot-set: personalRelationship \\ slot-set: score \end{cases}$$

Figure 30: Detailed Implementation of Rule 4 - Categorising Messages by Sender into Family Group

5.3.4.2 Key features2 - Context event

- Rule 5: Content event influence the score of message facts.

Description:

Messages that imply work (from time, location) or resting will influence the importance (score) of messages. The importance will be increased by different values depending on work or resting.

$$\text{R5: if } \left.\begin{array}{l} location \in work/rest \\ \wedge\, time \subseteq work/rest \\ \wedge\, other\_evidence \end{array}\right\} \Rightarrow \text{ slot-set: importance}$$

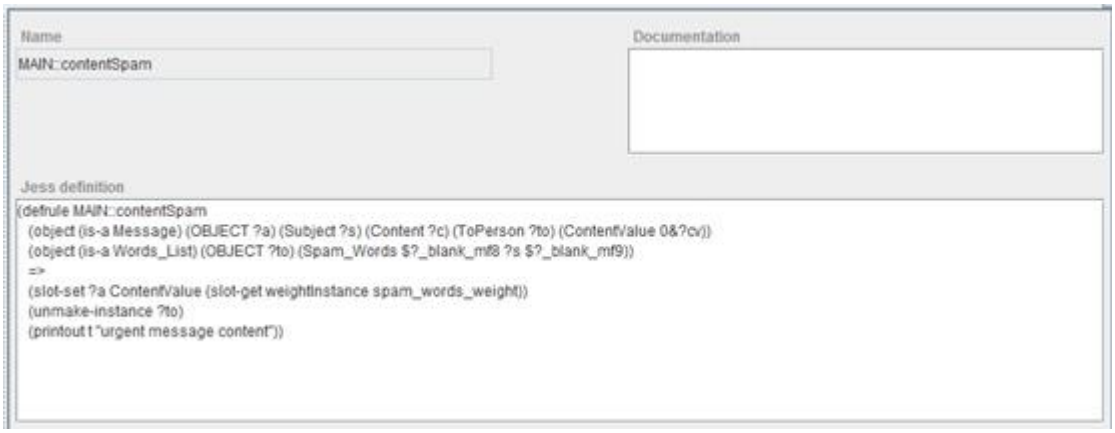


Figure 31: Detailed Implementation of Rule 5 - Categorising Messages by Its Content

- Rule 6: Context- someone approaching the user

Description:

When someone is approaching, the messages related to the person will be delivered.

$$\text{R6: If } \left.\begin{array}{l} approaching\_person \in Person \\ \wedge \sum Message(approaching\_person) \end{array}\right\} \Rightarrow \text{slot-set: importance}$$

Figure 32: Detailed Implementation of Rule 6 - Context Event - Someone Approaching

5.3.4.3 Key features3 - Handling messages of similar contents

- Rule 7: Upgrade similar messages

Description:

Messages of similar content will upgrade the importance (score) of message facts and remove duplicates.

$$\text{R7: If } \left.\begin{array}{l} m1 \leftarrow message(content) \\ \equiv m2 \leftarrow message(content) \end{array}\right\} \Rightarrow \begin{cases} slot-set: score \\ remove-instance: m1 \end{cases}$$

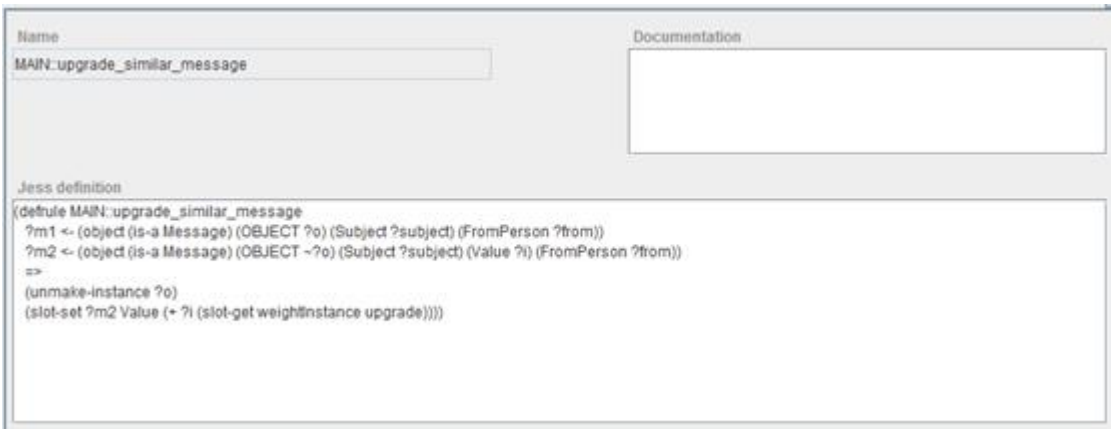


Figure 33: Detailed Implementation of Rule 7 - Upgrading Similar Messages

Executing and process of rules

This section will detect the process of activating facts and firing rules. Protégé provides tools to watch the detailed process of rules. We will demonstrate the implementation from overview to detailed ones.

The working memory includes all facts during run-time, where novel facts can be asserted and old facts can be transformed.

Facts in working memory are shown in Figure 34:

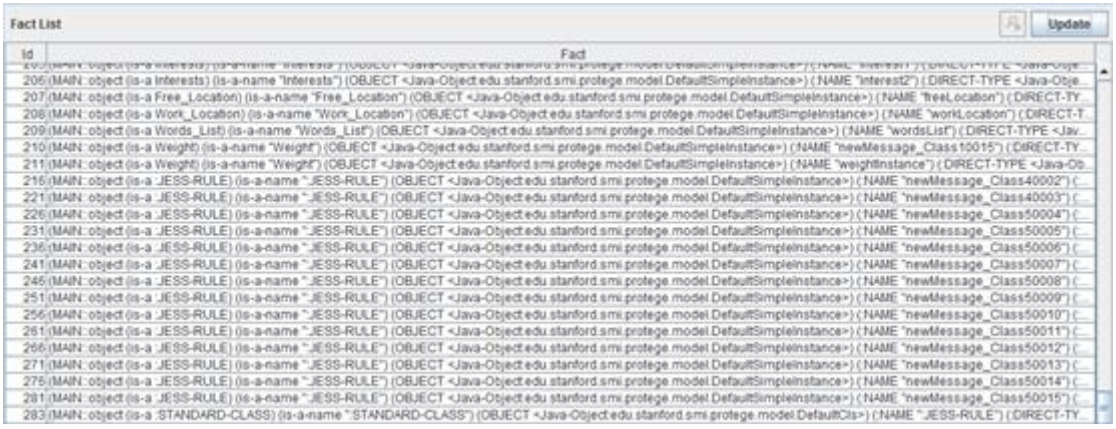


Figure 34: Screenshot from Protégé-All the Facts in the Working Memory

As we can see, there are large numbers of facts. However, even a single rule has to match every facts in the working memory. For the purpose of improved performance, it is recommended to adopt architecture 2 to divide facts.

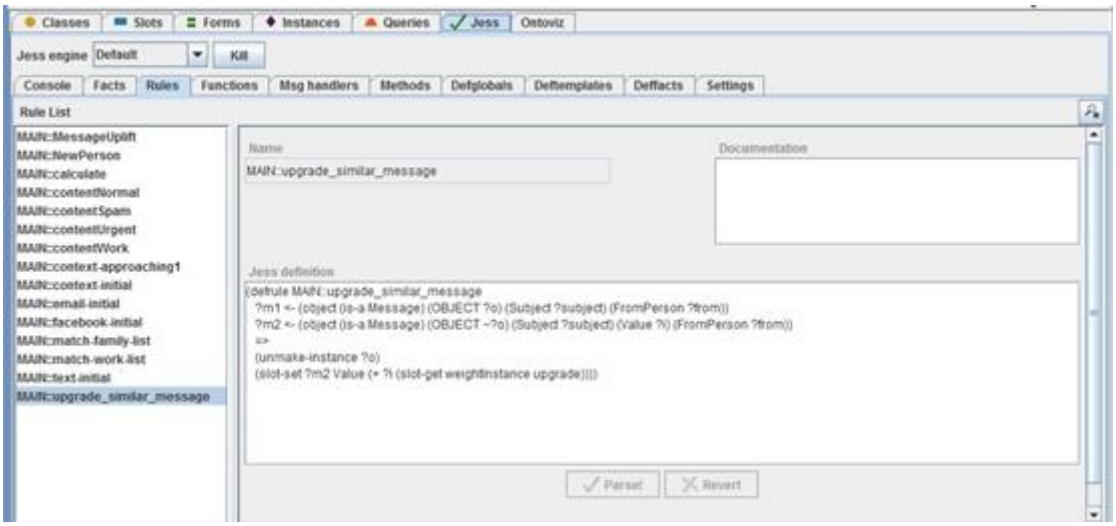


Figure 35: Screenshot from Protégé- All the Rules in the System

Figure 35 above demonstrates rule list related to use case 1

The executing process can be watched by watching rules via protégé as it shows in Fig 36.

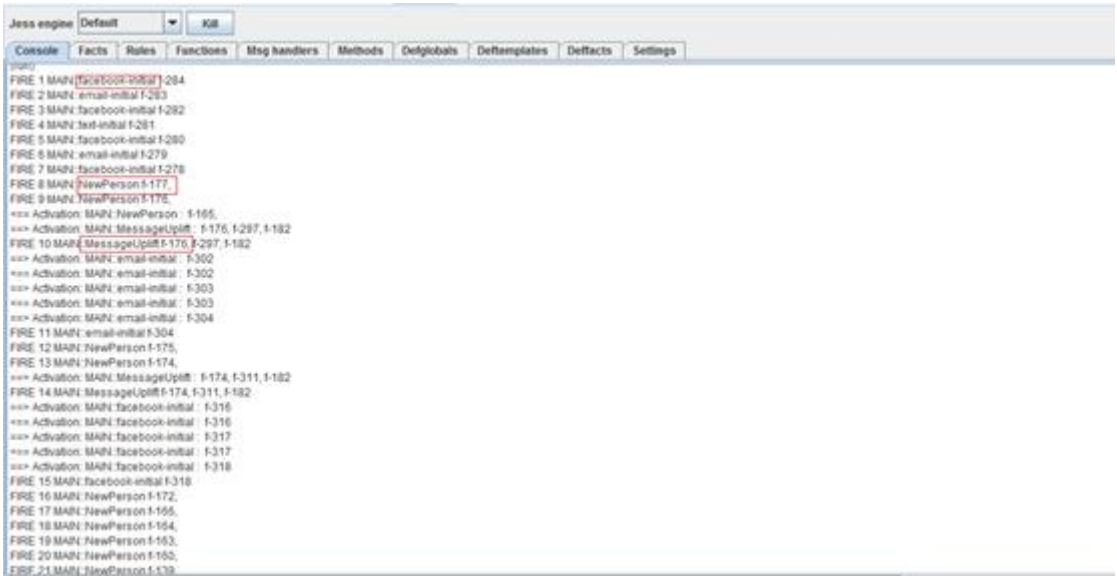


Figure 36: The Order of Processing Message A. Watching Facts In Compiling And Executing After All The Rules Are Added In The System.

Figure 37: The Order of Processing Message B. Watching Facts In Compiling And Executing After All The Rules Are Added In The System.

From Figure 36 & 37. It is evident that system randomly chooses rules to fire. Fig 36 fires rules in such order as facebook-intial, new person and message uplift while Fig 37 chooses a different order. Given a second thought, the approach can be improved by firing important rules at the first by giving priorities. For example, messages from boss are definitely delivered, so this rule can be fired at first then other rules about identifying contents or contexts will not be fired because the operation has already been determined.

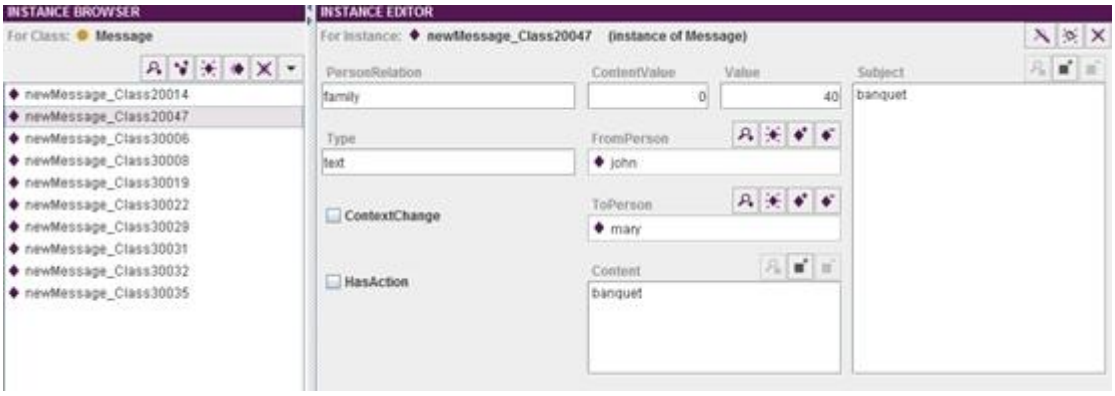


Figure 38: The Final Result after Executing Message A. Because The Value Is 40, Less Than 60, The Message Will Be Pending.

The running result is shown on Figure 38 where the content of messages is “banquet”, belonging to normal content, from family and text messages. It is an important message. However, the user is working now, the message will be in pending state until the context state get changed.

5.3.5 COMPARE OWL AND ONTOLOGY

This part will compare the reasoning capability between OWL with Jess and Protégé class with Jess. Owl and ontologies are different in many aspects, however, we will focus on reasoning aspect and take reasoning on personal relationship as an example.

As mentioned in the earlier section. In Protégé class, the personal relationship is object-based. If a person is a member of family, the person will be added into the family list of the user using multi-slot. The reasoning process is also accomplished by searching such person in the family list.

In comparison, OWL has rich library and better reasoning capacity, which provides a tree-like mechanism similar to object oriented. This part will implement a simple version of message assistant which is used to reason if a person is from family. With this example, we can compare OWL and ontology.

5.3.5.1 Design and Implementation of OWL

Protégé OWL plugin has been used to define an OWL ontology describing the class, Person, interests and contexts, and relationships, isFamily and isWork as shown in figure 40 and 41. In order to reason that a person is from family, a SWRL rule will represent some dependencies edited with the Protégé OWL plugin.

$$Family := isFamily \cup Person$$

Owl codes VS Ontology codes

```
<owl:Class rdf:about="#Person">
  <owl:disjointWith>
    <owl:Class rdf:ID="Interests"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Context"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="MessageOrigin"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Family">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Email</rdfs:label>
  <owl:disjointWith>
    <owl:Class rdf:about="#Other"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Work"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:SymmetricProperty rdf:ID="isFamily"/>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Person"/>
```

```
([john] of  Person

    (Account "john.smith@gmail.com")
    (Age 0)
    (Email "john.smith@gmail.com")
    (Family_List
        [mary]
        [john])
    (Name "john")
    (Other_List [facebook])
    (PhoneNumber "0834665525")
    (Work_List [john]))
```

Figure 39: Owl-Family design Vs Ontology-Family design

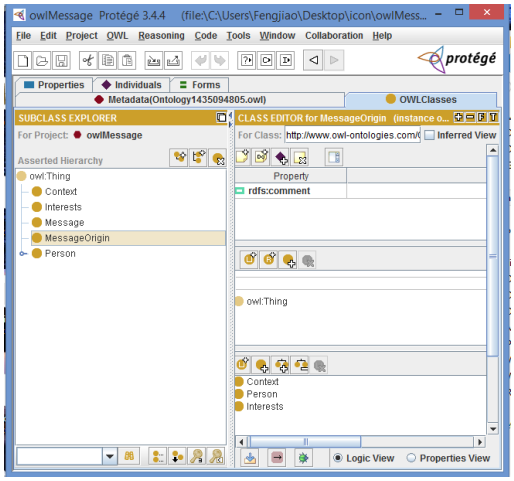


Figure 40: OWL-class

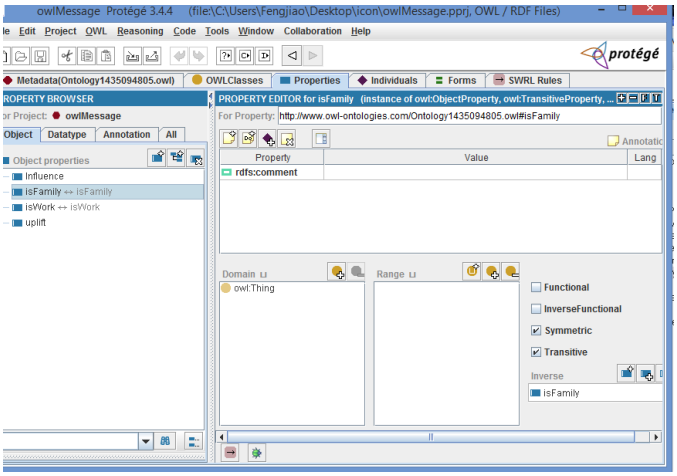


Figure 41: OWL-properties

5.3.5.2 The Comparison Between OWL and Ontology

OWL is well suited to represent "structured" knowledge by classes, properties, and taxonomies. SWRL is suited to deduce knowledge by rules composed of atoms. From this example, the family relationship is easy to reason by SWRL.

In comparison, reasoning with ontology in this application, relies on keyword list. Jess engine searches the family members list which requires update the family_list in run-time. The drawback is a waste of storage and matching time.

Besides, as we can see Figure 39. Ontology is simpler than owl but one family list has to be redesigned for every person ontology.

5.4 IMPLEMENTATION OF USER CASE 2 - ANNA

Even though use case 2 is different from use case 1 in many aspects, the internal challenges are similar. For instance, message categories and context handling, etc. The standard of the importance of messages will be transformed depending on user preference.

This part will give a review of use case 2 and identify the key challenge as well. Then it will describe the handling process of use case 2 as well as rules. Given that many challenge has been solved in the use case 1. Use case 2 will focus on handling dynamic user interests.

5.4.1 CHALLENDGES REVIEW OF USE CASE 2

Anna is a research student. Her work time is flexible. Her activities in a day include doing research, as well as engaging her interests. Her feature is dynamic interests and flexible working time. It is assumed that the place she stayed is major sign of doing research.

Key challenges

Given that many challenges have been solved in the use case 1. Use case 2 will focus on handling dynamic user interests.

The user is able to transform the importance of interests and assert novel interests into the working memory.

In addition, the importance of such interests can be changed with user involvement.

5.4.2 IMPLEMENTATION OF ONTOLOGICAL MODELLING-INTEREST

Interest is a class to describe the interest that the user hold, it has values to present the level of the interest. Most importantly, the value of interest level can be transformed. The implementation detail of related rules will be described in the use case 2.

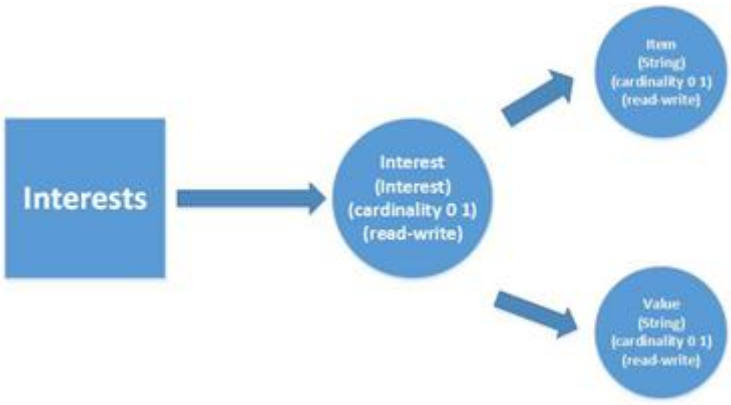


Figure 42: Graphic View of Interests and Interest

Figure 43: Protégé-An Instance of Interests.class Is Interest.class

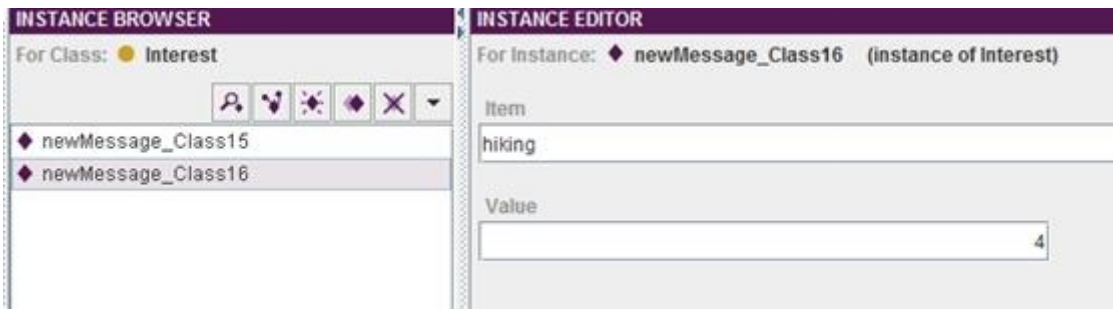


Figure 44: Protégé - An Instance of Interest Contains Preference Value and Interest Name

Fig 43&44 illustrates the instance of Interests. Two instances are detected and both of them contains a slot of interest type. For an interest instance as shown in Fig 44, the interest of hiking keeps a changeable value.

5.4.3 IMPLEMENTATION OF RULE ENGINE

Rule 8: Upgrade messages

Description:

If the message contains keywords about user's current interests, the importance of the message will be upgraded.

R8: If $\left.\begin{matrix} Content \in Interest \\ \wedge\ Interest \in Interests \end{matrix}\right\} \Rightarrow$ slot-set: importance

```
(defrule calclate
(object (is-a Interests)
(OBJECT ?interest)
(Interest1 ?v))
(object (is-a Message)
(OBJECT ?message)
(value ?v2))
=>
(printout ?t "caluate value " calculateall( (slot-get ?interest value) (slot-get ?message value))))

(deffunction calculateall(?a ?b)
(+ ?a ?b)
)
```

Figure 45: Implementation Detail of a Rule Which Is To Upgrade a Message Containing User's Interests

Figure 46: Assert the Rule into Protégé

The Figure 45&46 illustrates the detailed implementation of rules handling interests. The steps are identifying interests, and upgrade the importance of messages.

- **Rule 9: Transform the standard of importance**

Description:

In order to let system leverage user's current interest and habit, the value of weight standard should transform.

R9

```
setValue
(make-instance weightInstance of Weight)

;;(defrule update-weight
;;(object (is-a Weight)
;;(OBJECT ?weight))
;;=>
;;(slot-set weightInstance email_weight fuzzy() )
;;)
```

Figure 47: Rules for Updating Importance Standard

The running result in Fig 47:

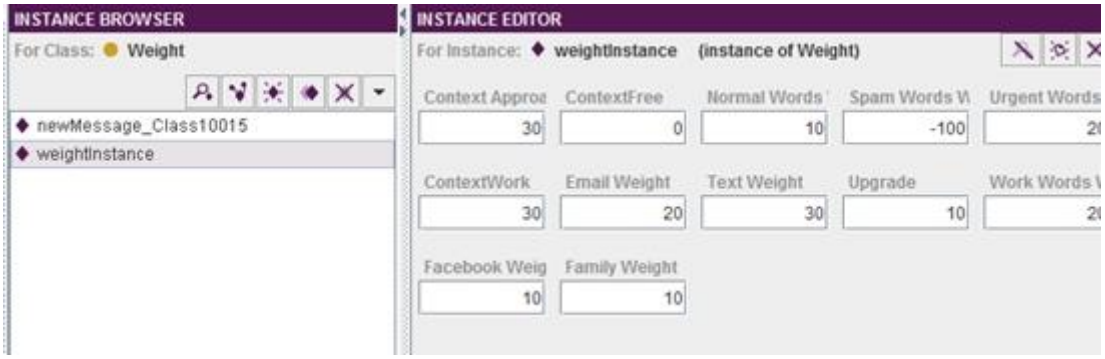


Figure 48: The Screenshot for Updating Weights/Importance Standard

Chapter-6 EVALUATION

6.1 INTRODUCTION

The design and implementation of the system are in an iterative process distributed in the following experiments and they are all needed to be evaluated from different aspects, like performance, feasibility, functionality, etc. This chapter describes the overall evaluation to examine the effectiveness of the rule-based approach, limitation and boundary of the research in this dissertation.

6.2 EVALUATION STRATEGY

The evaluation strategy intends to validate the personalised ontology and rule-based approach applied on the message assistant, as well as the effectiveness of ontological information representation and rules.

The iterative evaluation experiments described in this chapter are used to provide evidence that rule-based approach has the limitations and advantages from the perspective of functionality, performance, and fault tolerance to handle messages. In addition, architectures are evaluated as well to compare the facts involved, activated, and rules fired from the perspective of performance, load balance, etc.

The system has two use case containing messages description, context description and user profile description. We put forward series experiments to evaluate the system from limited facts and rules centred on use cases, to large numbers of facts and rules generated by the message simulator, to novel messages generated by Java Mail API. Three main experiments require comparison on activation numbers and rules fired order during compilation and execution. For the purpose of evaluation, the correctness, load, and performance were analysed. The experiments are listed as followed in session 6.3.

6.3 ITERATIVE EXPERIMENTS

From the general overview, the experiments were divided into three stages refer to Figure 49. The first stage is centred on use case described before. This stage will make a detailed analysis on the executing process. Take use case 1 as an example, in Architecture 1. We analyse the incoming messages, and context, user profile as well. Factors contributed to the operational

decisions are extracted. We will use logic algorithms to analyse the expected results and actual results.

As to architecture comparison, we will compare the numbers of facts, the fact activation, and rules. As it turned out, the result is corresponding to theoretical analysis in chapter 3. At the end of this part, a new architecture is raised and given theoretical analysis.

The second case focuses on testing the scalability and correctness of this system. Thus, a message simulator which can generate conditional messages is required. This java application will access messages stored in XML file and send to protégé system like the picture - stage 2;

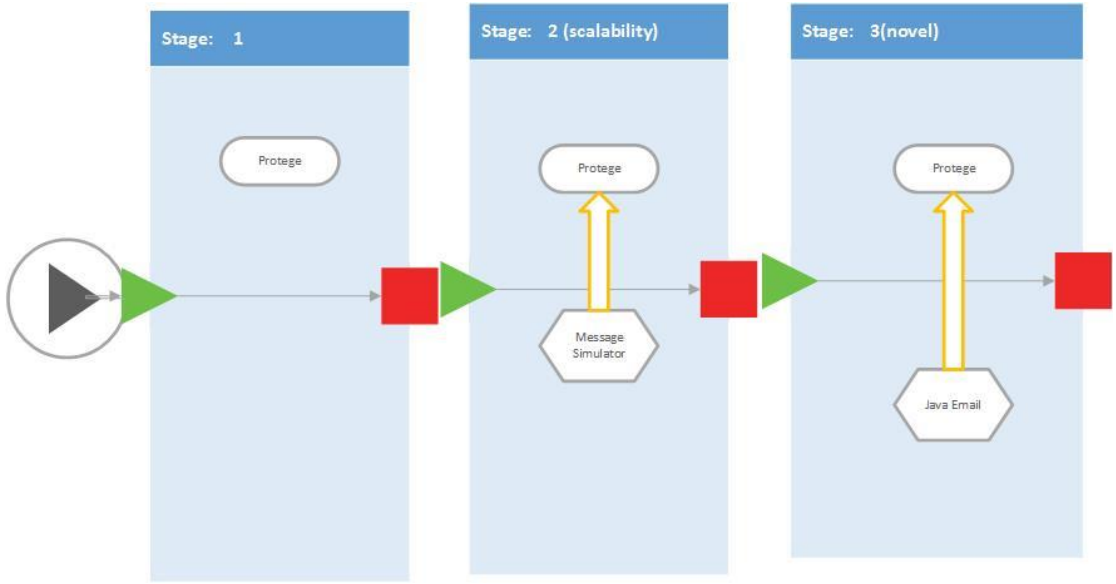


Figure 49: Evaluation Process- Three Stages

This stage will follow the similar evaluation pattern like stage 1. But it will carry on operation correctness evaluation (if the message will get the correct operations, delete/deliver/pending) because it is unnecessary to compare the correctness of importance of messages.

Stage 3 will focus on evaluating whether the system can handle novel messages, and novel people and enlarge the library of the system; that is the flexibility and fault tolerance of the system.

To conclude, the evaluation plan is listed in table 1. This procedure can be summarised as a performance evaluation to see if rule-based approach is a feasible approach.

Table 3: Purposes and Employed Architectures of the Evaluation Process

	Experiment	Message Source	Architecture	Evaluation focus
Stage 1	Exp. A.1	Protégé (use case)	Architecture 1	Functionality
	Exp. A.2			
	Exp. A.3			
Stage 2	Exp. A.1	Java application (Message simulator)	Architecture 1	Scalability performance
	Exp. A.2			
	Exp. A.3			
Stage 3	Exp. A.1	Java application (Java Mail API)	Architecture 1	Fault-tolerance(novel messages) flexibility
	Exp. A.2			
	Exp. A.3			

6.4 ITERATIVE EXPERIMENT – STAGE 1

Stage 1 centred on use case, with limited numbers of messages, context information, and limited user interests. This section will test the functionality of the system, and evaluate the correctness using mathematic statistic tools. To begin with, six messages are analysed and given the importance score based on importance standard. Then the process of rule execution will be given by flow chart with detailed descriptions about performance and cost during runtime. In the last part of this section, a mathematic analysis is given which explained the feasibility and effectiveness of the rule-based approach in handling messages.

6.4.1 NINE MESSAGES AND THEIR EXPECTED RESULTS

Context:

- Work 1:00 p.m. – 3:00 p.m.
- Resting 3:00 p.m.-5:00 p.m.

Personal Interests:

- Football

Message

- Message 1: 1:00 email from Mary (wife) about banquet tonight at 9 p.m.
- Message 2: 1:10 text from a co-worker (sue) about company news.
- Message 3: 2:00 email about urgent meeting (from secretory) right now for 20 min.

- Message 4: 2:10 text from friends (Sunday afternoon).
- Message 5: 2:10 text from co-works about a work plan.
- Message 6: 2:30 text from Mary about banquet tonight at 9 p.m.
- Message 7: 3:10 Facebook about friend request.
- Message 8: 3:20 Facebook about a football game.
- Message 9: 3:30 Facebook about music events.

Weighing Standard.

Note: the importance standard can be transformed to fit user's preference. This one is system default.

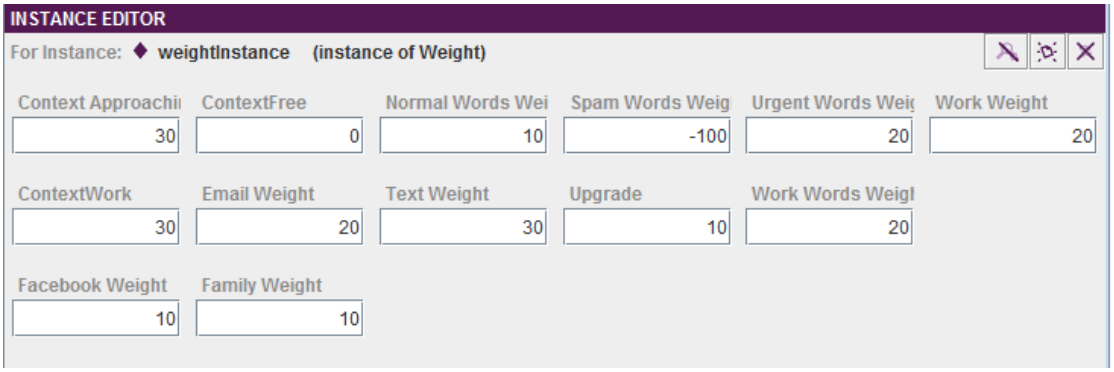


Figure 50: Default Weight Standard

Referred to Figure 50, weighting is listed in the table. When someone approaches, the current importance of the message increased by 30. When the user is at work, the messages will get increased by 30 as well. However, worth noting is that if the message contains spam work, the message's importance will be less than 0. Thus, it is deleted directly.

Table 4: Analysis of Messages

	Work	**sender**	**content**	**Urgent?**	**Similar**	**Interest**	**Actual (Y)**	**Expected (Ye)**
m1	Y	family	normal	N	N	N	40(0)	40(0)
m2	Y	work	work	N	N	N	60(1)	70(1)
m3	Y	work	urgent	Y	N	N	60(1)	70(1)
m4	Y	friend	normal	N	N	N	30(0)	30(0)
m5	Y	work	work	N	N	N	60(1)	60(1)
m6	Y	family	normal	N	Y	N	60(1)	60(1)
m7	N	unknown	spam	N	N	N	-90(-1)	-90(-1)
m8	N	unknown	normal	N	N	Y	70(1)	70(1)
m9	N	unknown	normal	N	N	N	40(0)	40(0)

Note: In the expected column, 1 means the message should be delivered, 0 means pended until context event happened, -1 means that message should be deleted.

The standard set that if the importance of messages is less than 0, the message will be deleted; if the importance of the messages is more than 60, the message will be delivered in no time.

The procedure of rule handling is displayed in Figure 51:

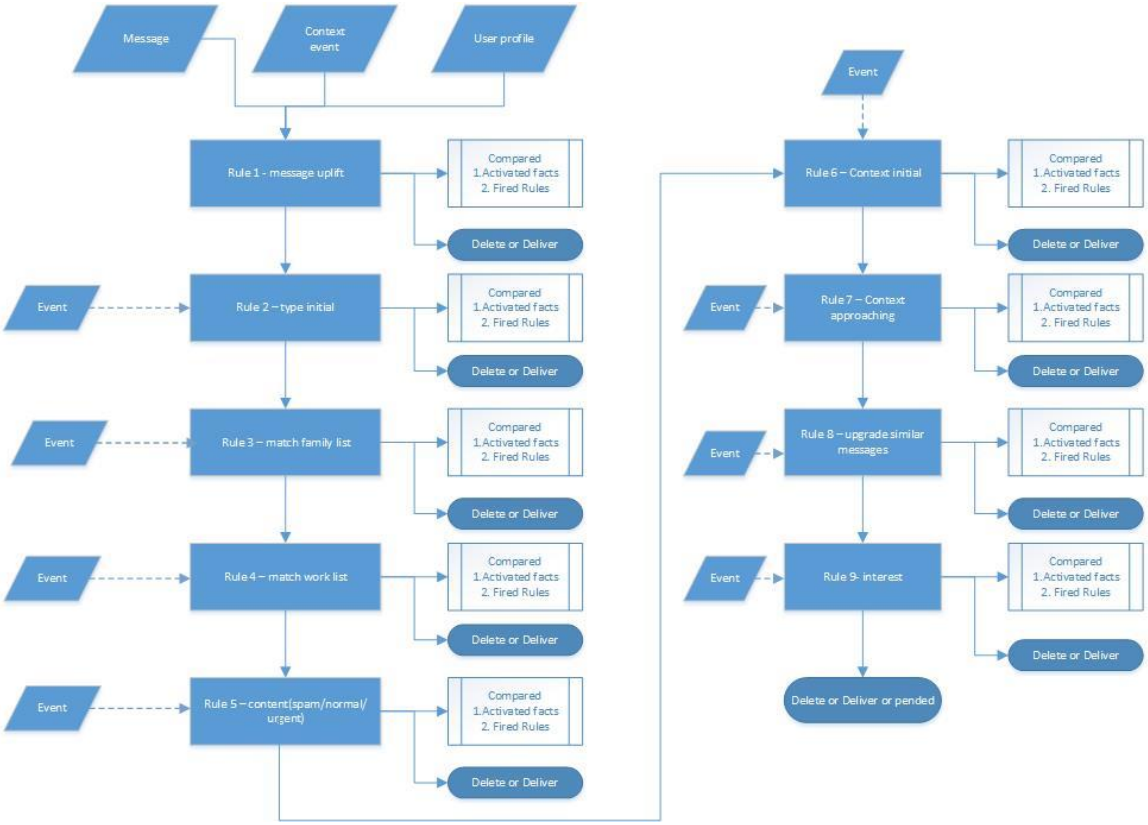


Figure 51: Detailed Procedure of Handling Rules. However, The Order Of Processed Rule Is Random Depending On Rule Engine.

In Fig 51, the process of handling message is given. However, the order of firing rules not necessarily follows the order as it shown in the Figure 48. For every rule, some facts (or instance) are activated in any order. One of the intentions of evaluation is to analyse the number, and order of activated facts and fired rules.

These facts are activated following a random order. The result are "watched" with the help of protégé.

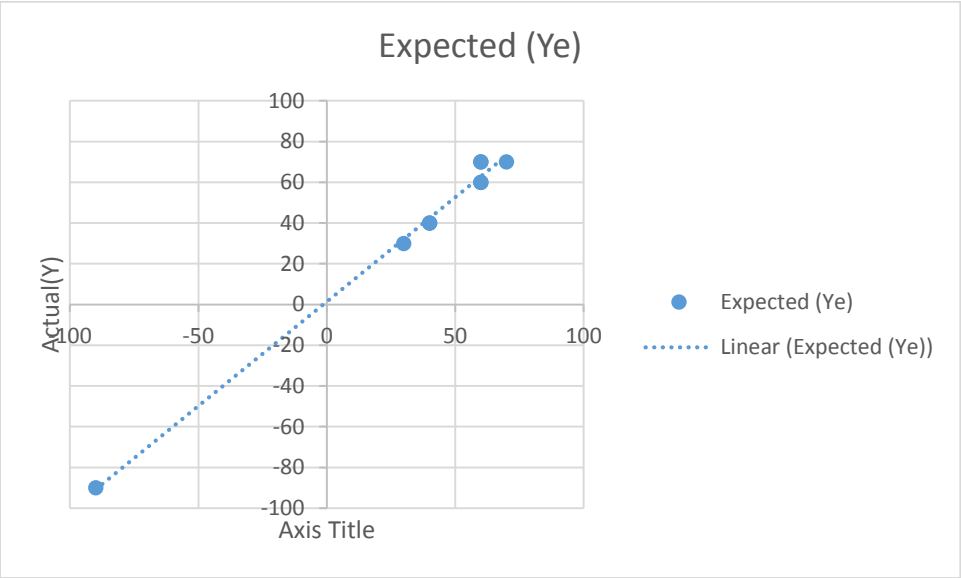
- R1: 17 facts are activated in compilation, rule are fired once in execution.
  Total: 166 facts.
- R2: 27 facts are activated in compilation, rules are fired 9 times in execution.
  Total: 241 facts.
- R3: 9 facts are activated in compilation, rules are fired 9 times in execution.

Total: 241 facts. (R3 combines 3 sub-rules, email-initial/facebook-intial/text-initial)

- R4: 10 facts are activated in compilation, rules are fired 5 times in execution.
  Total: 285 facts.
- R7: 4 facts are activated in compilation, rule are fired once in execution.
  Total: 369 facts.

In total, 379 facts are activated and 7 rules are fired. The usage of memory is the expected number of activated facts are, relevant to the number of incoming messages.

The running result compared with expected result are analysed using regression analysis:



SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.996618
R Square	0.993246
Adjusted R Square	0.992121
Standard Error	4.802124
Observations	8

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	20349.14	20349.14	882.4281	9.65E-08
Residual	6	138.3623	23.06039		
Total	7	20487.5			

RESIDUAL OUTPUT

Observation	*Predicted 40*	*Residuals*
1	63.08188	6.918117
2	63.08188	6.918117
3	32.34687	-2.34687
4	63.08188	-3.08188
5	63.08188	-3.08188
6	-90.5932	0.593166
7	73.32689	-3.32689
8	42.59188	-2.59188

Conclusion: as we can see from the table, the R square is above 95%. So the experiment result is highly acceptable. It means that rules can help give precise operations on incoming messages.

6.5 ITERATIVE EXPERIMENT – STAGE 2

After the analysis in Stage 1, it is clear that even a single rule can influence all the facts in the working library, even though only a small number of facts are activated in a random order. When the system is scale to handle a large number of messages, and complex context information, performance, accuracy and system load can get worse or keep stable like stage 1. This section will analyse many situations where the number of messages changes, urgent number transforms, and other selection conditions changes.

This stage will use message simulator, a java-based application to generate message meeting the selected conditions.

The experiments are conducted to test the effectiveness of rules

6.5.1 EXPERIMENT B

Test rule 4: The effectiveness of rule 5 which handles messages from family when the number of family ones get changed.

Description: The total number of messages, the time period, and the urgent number keep unchanged while the number from family is changed from 20, 50 to 100.

The randomness of messages can be get from changing the Start_time and End_time of messages.

The values in the test are in table 7 &8:

Table 5: Experiment 2.Stage 2 - Parameters Table

Total number:	200
From family:	X
Start time:	variable
End time:	variable
Urgent number:	20

Table 6: Experiment 2.Stage 2 - Experimental Result

Family(X)	expected	actual	facts	rules
20(200)	20(125)	20	1234	164
50(200)	50(154)	42	1290	168
100(200)	100(108)	44	1010	136

The correctness's level shows in Fig 52-54:

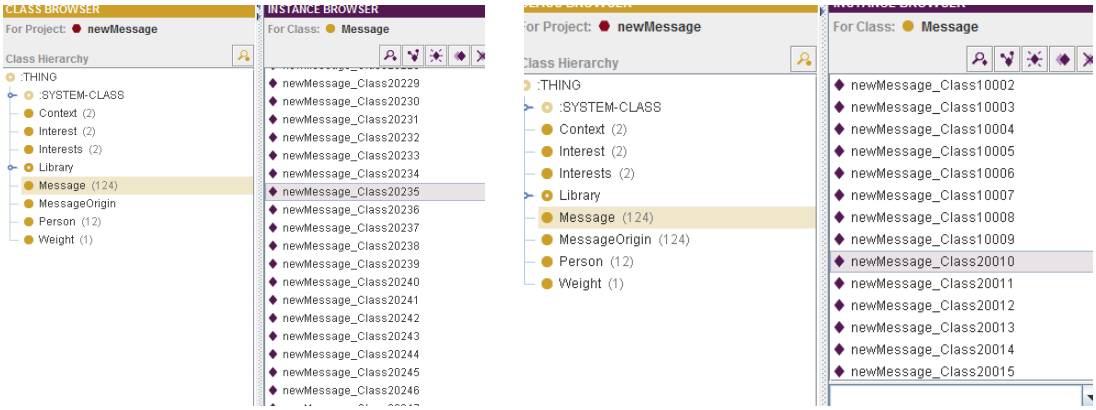


Figure 52: The Family Number Is 20, Before Employing Rules And After Employing Rules

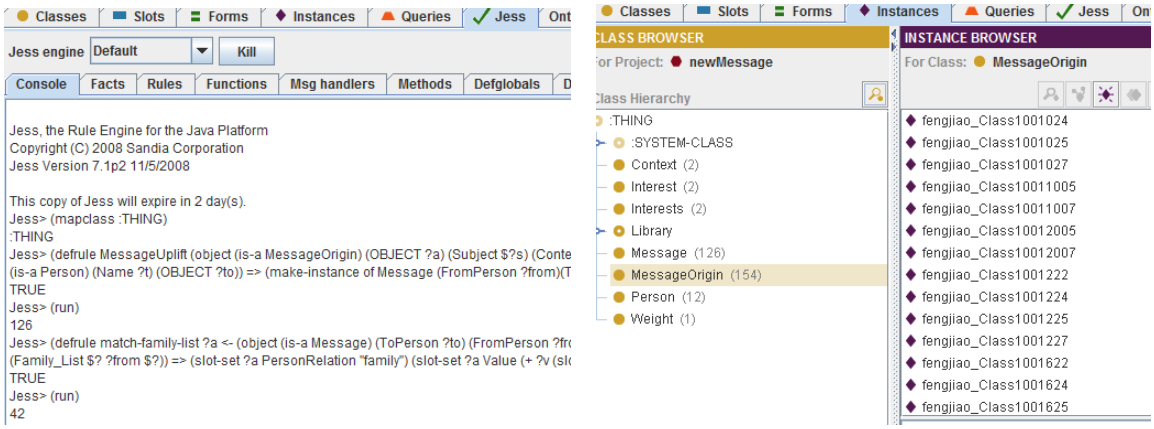


Figure 53: The Family Number Is 50, Fired Facts Which Belong To Family Is 44

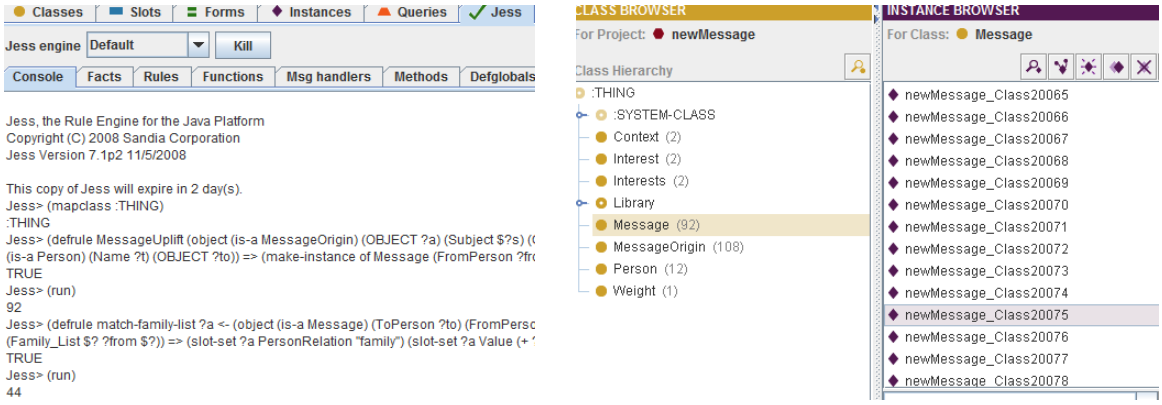


Figure 54: The family Number is 100(200), however, only 92 messages were inserted while 44 belongs to family.

To conclude, as we can see from the Figure 52-54, the correctness of the system is not good, while the performance is basically acceptable. Many memories get waste under this architecture.

6.5.2 EXPERIMENT C

Test whole rules when the whole number of messages from family get changed.

Description: The urgent number of messages, the time period, and the number from family keep unchanged while the whole number is changed from 50, 100 to 200. Rule 2 and rule 7 are fired. The correctness should be that the number of non-replicate message is 23 no matter how many messages are imported into the system.

The randomness of messages can be obtained from changing the start_time and end_time of messages.

The values in the test are shown in Table 9&10:

Table 7: Experiment 3.Stage 2 - Parameters Table

Total number:	X
From family:	20
Start time:	variable
End time:	variable
Urgent time:	20

Table 8: Experiment 3.Stage 2- Experimental Result

Total(X)	expected	actual	Initial facts	facts	Fire rule
50	50	41	206	663	93
100	100	69	259	998	143
200	200	133	316	1645	291

The correctness's level shows in Figure 55-56:

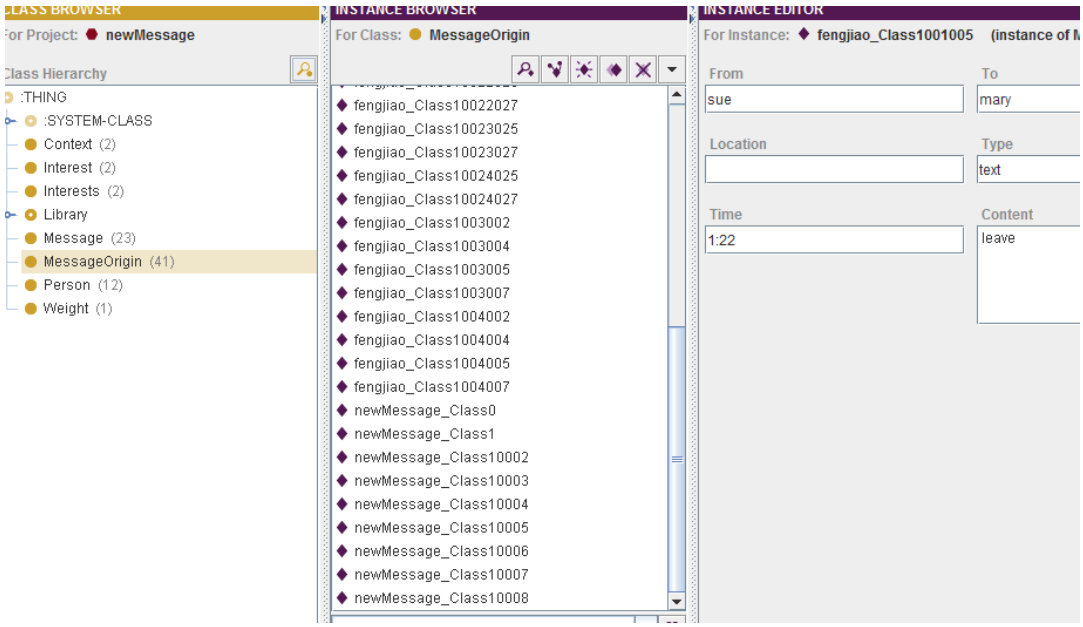


Figure 55: Total number is 50 when only 41 messages are inserted but the running result is correct.

The analysis of Fig 55: As we can see, the number of messages changes from 41 to 23, it is correct. But not all the messages are scanned and executed.

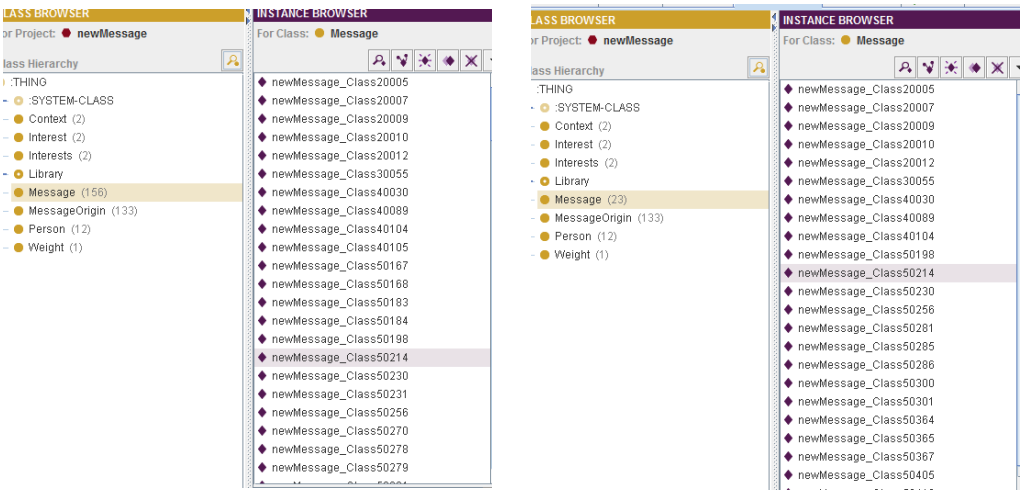


Figure 56: Total number is 200 when only 133 messages are inserted but the running result is correct.

The analysis of Figure 56: As we can see from, the number of messages changes from 131 to 23, it is correct. But not all the messages are scanned and executed.

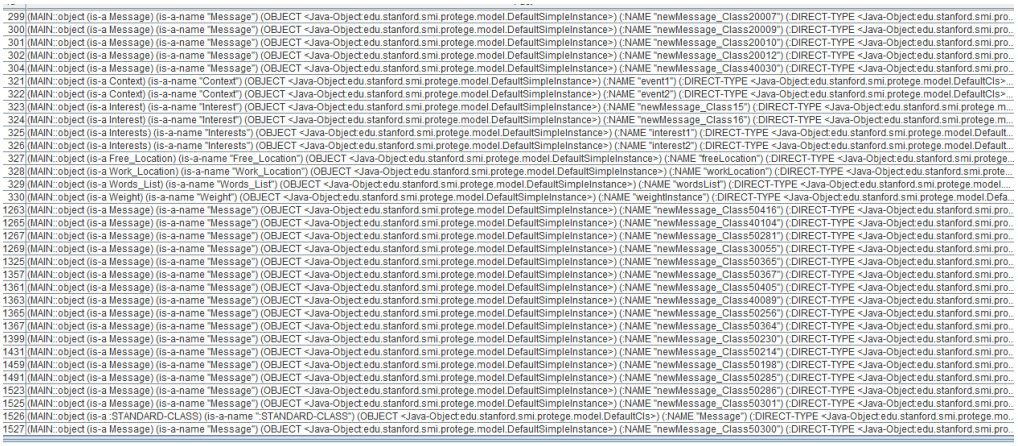


Figure 57: The number of facts in the working memory is huge

The response time of system is acceptable when there is a large number of messages. However, many facts exist in the working memory which increases the response time and cost.

To conclude, as we can see from the figure, the correctness of the system is comparably reasonable as well while the performance is not good. Many memories get waste under this architecture.

6.6 ITERATIVE EXPERIMENT – STAGE 3

After the experiments in stage 2 focusing on scalability and performance test of these rules, this part will evaluate the fault tolerance of the system when there are novel messages.

In order to generate novel messages, we designed and implemented a java application using JavaMail API, generating emails from the mailbox.

6.6.1 NOVEL MESSAGE GENERATOR

This generator aims to retrieve emails from the mailbox using Java Mail API and then uplift emails into metadata form. Importantly, it is necessary to uplift keyword list from subject and content of messages. For example. “I want an apple” will be uplifted into “apple”.

Given the challenge, we will design a wordhandler.class to uplift sentences.

Design of the message generator

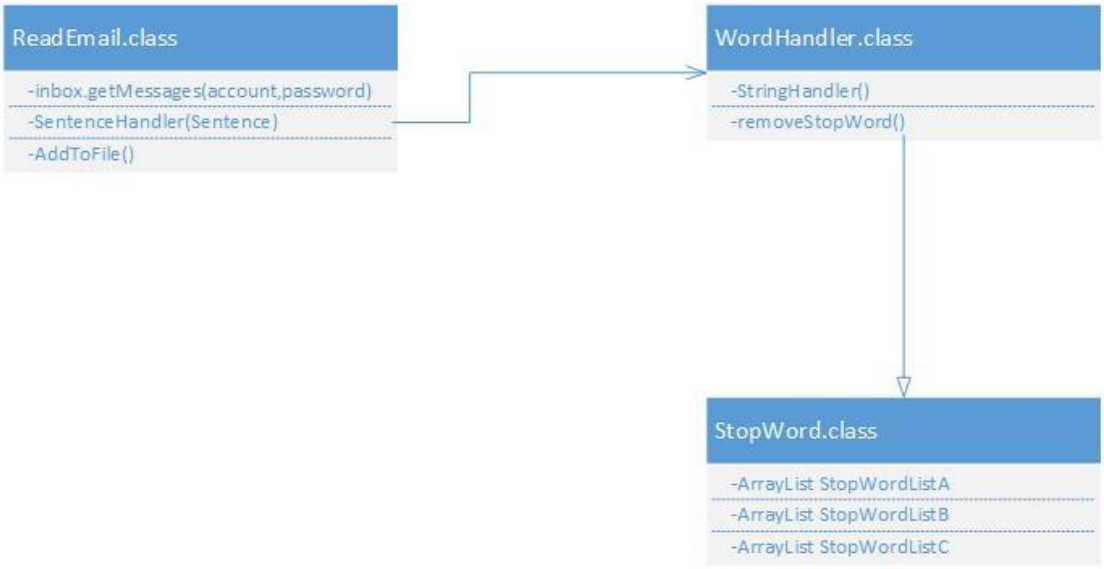


Figure 58: UML Class

1) ReadingEmail is responsible for retrieving raw email from mailbox.
2) Wordhandler focuses on extract keywords list from content and subject of emails.

Thus, the incoming messages from the mailbox, which can be composed of different contents, has lots of uncertainties. Thus, using mailbox meets our requirements of generating novel emails.

Implementation of Java Mail

With the help of JavaMail.API, message generator is able to connect mailbox supporting IMAP protocol and authenticate Gmail account and its password.

The executing result is displayed in Figure 59.

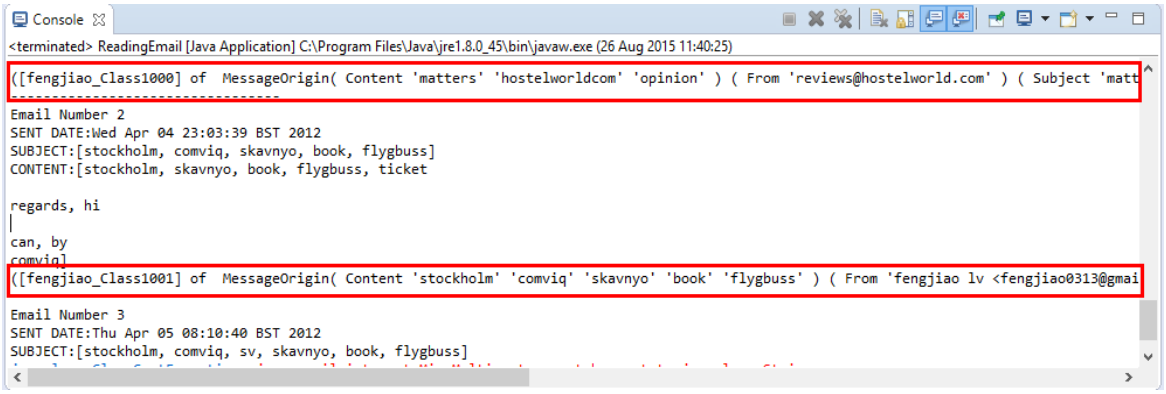


Figure 59: Generating Novel Messages

6.6.2 EXPERIMENT

With that, we choose three novel message is imported into protégé system.

- Message 1: From HostelWorld.com, an email about asking for their service.
- Message 2: From Convip, booking ticket to Stockholm via Ybuss.

The expected result should be that message 2 is delivered or pended because the content is important about confirming; message 2 should be deleted, because of unimportant advertisement;

Table 9: Experiment. Stage 3 - Experimental Result

	Expected Importance	Expected operation	Actual Importance	Actual operation
Message 1	<0	-1	20	0
Message 2	30	1 or 0	20	0

The running result are displayed in Figure 60-61:

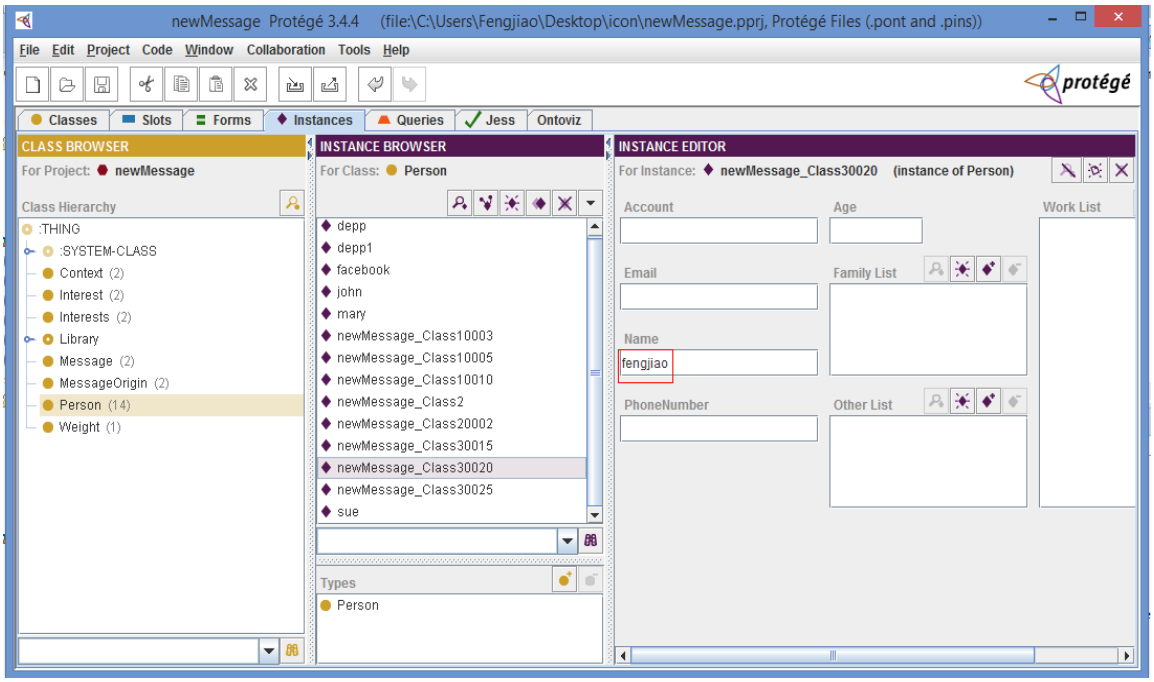


Figure 60: Creating New Users

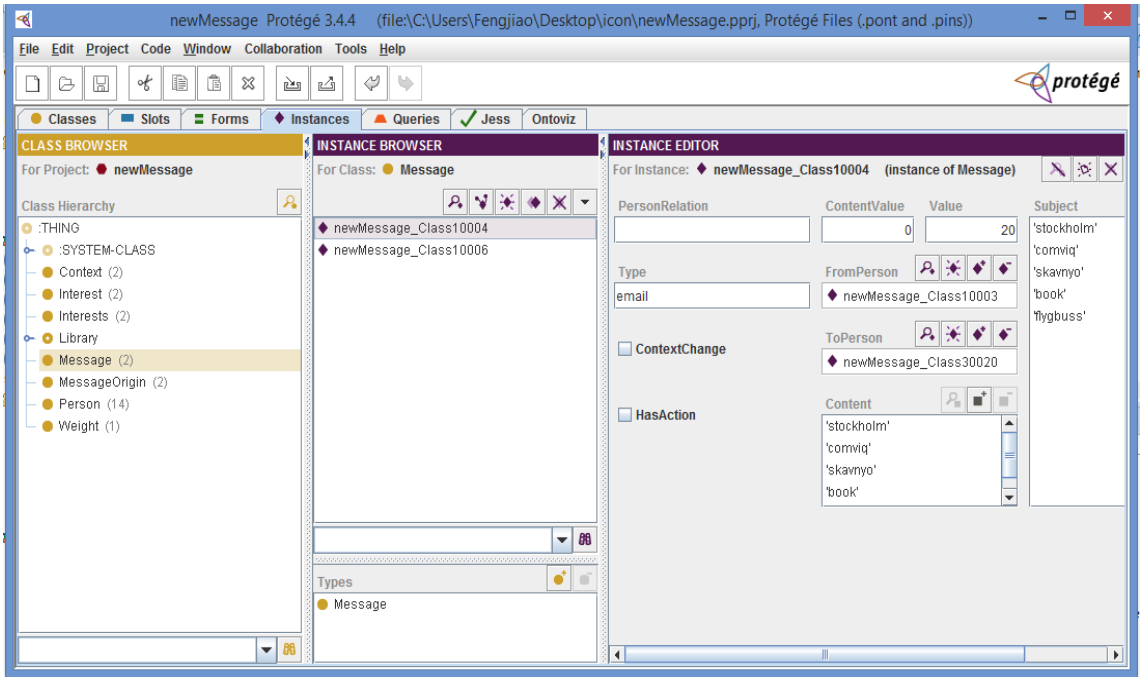


Figure 61: Generating Scores For Every Messages

Analysis: Because the keyword list fails to contain all the novel information. In this case, the system fails to identify the review as a sign of advertisement. So it takes every word that has not appeared before as normal word. It is a drawback of this method. The possible solution is to increase user scrutiny.

6.7 SUMMARY

In summary, the message assistant based on rule-based approach is able to fulfil functional requirements stated in chapter 1. However, the system fails to display good performance due to the inefficiency of Jess rule engine. With the increase of the message, operational decision becomes inaccuracy.

Chapter-7 CONCLUSION

The chapter presents the conclusion of the thesis. To start with, it illustrates an assessment of the extent to which the objectives of the thesis have been filled and the contributions that the thesis has made. Followed by the evaluation, the thesis discusses the limitations and future work.

7.1 ACHIEVEMENTS FOR RESEARCH OBJECTIVES

This thesis hypothesized that rule-based approach could be used to filter or hang incoming messages. To pursue this, a research question was set out as:

The research explores how and to what extend rule-based approach can be leveraged to make an operational decision on incoming messages, integrating context and user preference.

Five objectives listed in Chapter 1 are identified and solved:

Objective 1: Find out similar approaches to categorising and operating messages from the state of the art, and investigating issues surrounding rule-handling approach, and ontology modelling, etc.

Objective 2: Design and implement the information representation based on two stated two use cases, and evaluate this approach.

Objective 3: Design and implement the rule-based approach to handling incoming messages based on the two use cases. Then, evaluation and comparison based on the current approaches should be given.

Objective 4: In order to face the issue of massive messages, design and improve the general architecture and evaluate the architecture.

Objective 5: Follow an iterative approach to evaluating and validating the design and implementation of the design and implementation.

The state of the art reviews the current research about the rule-based approaches. Many existing pieces of research are addressed about information representations, semantic techniques and rule-based approaches. The discussion reveals that the current challenges are integrating dynamic context, user profile information with messages using rule reasoning. The state of the also illustrates some ways to reason based on limited knowledge, by combing heuristic-based approach with rule-based approach. Objective 1 is fulfilled here.

In Chapter Four, two architectures are described and analysed. Both of the architecture have advantages and disadvantages based on rule approaches and Jess engine. In particular, architecture 2 contains three subsystems, each of which focuses on handling related rules and related facts, improving the performance and cost of the system compared to architecture 1. Based on the analysis of Jess rule engine, architecture 2 can be improved, the improved architecture will be described in future work (see section 7.3.4). Objective 3 is fulfilled here.

Chapter Four and chapter Five gives detailed design and implementation. As it turns out, rule-based approach has the capability to decide operations on messages. Thus, it achieves the Objective 4.

Chapter Six is the evaluation and Objective 2 and 5 is fulfilled. The evaluation is approached as an iterative process during three experiments. The performance evaluation is accompanied with accuracy evaluation on a different data set. As it turns out, the current system fulfils the requirements. However, it has space to get improved.

7.2 CONTRIBUTION

The main contribution of this thesis is a rule-based approach, integrating dynamic context, and dynamic user preferences ontologies. This approach contributes to the state of the art by offering a distributed architecture and open-to-inspect system. Another contribution is that the system has realised some complex rules like upgrading similar messages, matching-content/interests precisely, etc. The other minor contribution of this system is the design and implementation of message assistant, a test-based applications to evaluate the effectiveness of such approach.

7.3 FUTURE WORK

From information representation to rule reasoning, from detailed implementation to architecture, there are numbers of practical and research issues in which there is potential for extensions and improvement to current work. The performance of the rule-based approach could be improved from following aspects.

7.3.1 INCREASE COMPLEXITY

Message complexity: In the current stage of information explosion, there are diverse communications via the Internet, like video, voice, ciphered messages, etc. It is required

integrated approaches to process these messages. In addition, as to complexity in messages, some messages may relate to other messages. If the message has to be delivered, other relevant pending messages earlier should be delivered as well.

Context complexity: To increase the accuracy of this message assistant, the following work can consider more complex contexts, like reasoning locations from a calendar, from close friends or from mobile location service.

Person complexity: User's needs can be complex in real life. A user may be interested in aunt's messages rather than nephew's message even though they both belong to a family group. In another case, a user may be interested in messages from a relative of aunt's, but such person is not in the family group. To make more complex, a simple interest can have multiple forms of representations. For example, swimming can be represented as the seaside, coach, bench, etc.

Examples above requires a flexible system with rich knowledge base which need us to explore proper technologies in future work.

7.3.2 REPLACE PROTÉGÉ ONTOLOGY WITH OWL

As demonstrated in chapter 3, OWL has a much better capacity to make semantic reasoning than protégé ontology. Thus, there is potential for extension to replace protégé class with OWL. OWL has better capacity because classes are organized into hierarchical structure. In addition, OWL has the ability to express rich semantic information by the triple, and provide semantic reasoning based on OWL information.

7.3.3 SETTING INITIAL VALUE

In the current implementation, some value are set based on user's input or system default value. However, it is suitable for use in real life. The importance of user's interest can be set based on historical learning based on past user's behaviour. However, in this system, we failed to provide such mechanism.

7.3.4 IMPROVE THE EFFICIENCY OF REASONING

Another way to improve the performance of the system is to improve the efficiency of rule reasoning. As the evaluation showed, the processing time took long time to complete a decision. A solution is to choose a more efficient rule engine and optimise the reasoning approach. Future work will explore current rule engine and compare the performance.

As to improve the reasoning approach. We can design the architecture and the rules. Architecture 2 has already improved the efficiency, but it can make better. The efficiency can also be realised by adding priorities to the given rules. If the preferred rule is fired at first, it has a higher possibility that other rules will not have to be fired in the later stage.

The reasoning approach can be optimised for using better in working memories and real-time reasoning approach.

7.3.5 ENLARGE THE KEYWORD LIST

Currently, the system contains a limited number of keywords. However, in real life, this is not enough. OWL has its library, which can be useful to adopt RDF/OWL to overcome such issues. Another solution may be that whenever a novel message comes, the related words will be imported into the library. Based on the user behaviour, the system begins learning if the word should be important or less important. Future work will focus on machine learning in response to limited knowledge and semantic information.

7.3.6 HETEROGENEOUS CONTEXT

Even though there are some approaches to handling heterogeneous context and context integration, the system just realised a simple version without considering complex context and its diversity. In real life, it is practical that lots of evidence of location, time are coming from different sources. It is recommended to adopt OWL to represent and reason the context information. In future work, we will import complex context information into the message assistant system.

7.3.7 SCALABILITY

With the advance of a rule-based system, many rules can be executed under different environments. Being cross-platform and easy to scale become another challenge for such system. The system should explore methods to handle an increasing number of messages and analyse security, performance, etc.

REFERENCES

[1] *Adaptive Hypermedia and Adaptive Web-Based Systems*. 2006: 4th International Conference.

[2] Doreen Cheng, H.S., Swarooop Kalasapur, Sangoh Jeong, *Situation-aware User Interest Mining on Mobile Handheld Devices.*

[3] Judy Kay, W.T.N., *PERSONAF: framework for personalised ontological reasoning in pervasive computing.* 2008.

[4] Jian Zhou, L.M., Qiaoling Liu, Lei Zhang, Yong Yu, Yue Pan, *Minerva: A Scalable OWL Ontology Storage and Inference System.* 2002.

[5] Gerhard Goos, J.H., and Jan van Leeuwen, *User Modeling,Adaptation,and Personalization*. 2013.

[6] Kay, J., *Scrutable Adaptation: Because We Can and Must.* 2006.

[7] Shen, J., Oliver Brdiczka, and Juan Liu, *Understanding email writers: Personality prediction from email messages*, in *User Modeling, Adaptation, and Personalization*. 2013, Springer Berlin Heidelberg.

[8] Wasinger, R., James Wallbank, Luiz Pizzato, Judy Kay, Bob Kummerfeld, Matthias Böhmer, and Antonio Krüger, *Scrutable user models and personalised item recommendation in mobile lifestyle applications.*, in *User Modeling, Adaptation, and Personalization*. 2013.

[9] Kelly, D., *Implicit Feedback for Inferring User Preference: A Bibliography.* 2003.

[10] Kay, J., *Stereotypes, student models and scrutability* 2006.

[11] O'Connor, A., *Context-Informed Semantic Interoperation*, in *Knowledge and Data Engineering Group,*. 2010, University of Dublin, Trinity College.

[12] Klyne, G., and Jeremy J. Carroll, *Resource description framework (RDF): Concepts and abstract syntax.* 2006.

[13] McGuinness, D.L., and Frank Van Harmelen, *OWL web ontology language overview.* W3C recommendation, 2004.

[14] song, y., *knowledge-driven information uplift approach to support non-expert users in monitoring and understanding network systems*. 2013, University of Dublin, Trinity College.

[15] Decker, S., Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks, *The semantic web: The roles of XML and RDF*. 2000.

[16] Matheus, C.J., Kenneth Baclawski, Mieczyslaw M. Kokar, Jerzy J. Letkowski, *Using SWRL and OWL to capture domain knowledge for a situation awareness application applied to a supply logistics scenario.*, in *Rules and Rule Markup Languages for the Semantic Web* 2005. p. 130-144.

[17] Sirin, E., and Bijan Parsia, *SPARQL-DL: SPARQL Query for OWL-DL.* 2007.

[18] Knublauch, H., Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen, *The Protégé OWL plugin: An open development environment for semantic web applications.* The Semantic Web–ISWC, 2004.

[19] Horridge, M., Holger Knublauch, Alan Rector, Robert Stevens, and Chris Wroe, *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0.* 2004: University of Manchester

[20] Eriksson, H., *The JESSTAB approach to Protégé and JESS integration." Intelligent Information Processing.* 2002.

[21] Eriksson, H., *JessTab Manual*.

[22] Gerhard Goos, J.H., and Jan van Leeuwen, *Rule and Rule Markup Languages for the Semantic Web*. 2004.
[23] Gerhard Goos, J.H., and Jan van Leeuwen, *Rules on the web: research and applications*. 2012.
[24] Platt, J., *Rule-Based System*.
[25] Thanyalak Rattanasawad, K.R.S., Marut Buranarach, Thepchai Supnithi, *A Review and Comparison of Rule Languages and Rule-based Inference Engines for the Semantic Web.* Computer Science and Engineering Conference (ICSEC), 2013.
[26] Friedman-Hill, E., *Jess® The Rule Engine for the Java™ Platform*. 2008.
[27] Friedman-Hill, E., *Jess in action*. 2003: Manning Publications Co.
[28] Harold Boley, S.T.a.G.W., *Design Rationale of RuleML: A Markup Language for Semantic Web Rules.* 2001.
[29] Horrocks, I., Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, *SWRL: A Semantic Web Rule Language Combining OWL and RuleML.* 2004.
[30] Christopher J. Matheus, K.B., Mieczyslaw M. Koka, *BaseVISor: A Triples-Based Inference Engine Outfitted to Process RuleML and R-Entailment Rules.* 2006.
[31] Benjamin N. Grosof , M.D.G., Timothy W. Finin, *SweetJess: Translating DamlRuleML to Jess.* 2002.
[32] Bolchini, C., Carlo A. Curino, Elisa Quintarelli, Fabio A. Schreiber, and Letizia Tanca., *A data-oriented survey of context models.* 2007.
[33] Elena Vildjiounaite, O.K., Vesa Kyllönen, and Basilis Kladis, *Context-Dependent User Modelling for Smart Homes.* Springer-Verlag Berlin Heidelberg, 2007.
[34] Victoria Bellotti, K.E., *Intelligibility and Accountability: Human Considerations in Context Aware Systems.* 2001.
[35] Mobasher, B., *Context-Aware User Modeling forRecommendation*. 2013.
[36] Bajwa, S.M.a.I.S., *A Rule Based Approach for Business Rule Generation from Business Process Models*
[37] Hans Weigand, A.P., *The Pragmatic Web: Putting Rules in Context.*
[38] Pankowski, T., *Using Data-to-Knowledge Exchange for Transforming Relational Databases to Knowledge Bases.*
[39] Stefano Bragaglia, F.C., Paola Mello, and Davide Sottara, *A Rule-Based Calculus and Processing of Complex Events.* 2012.
[40] Theodore Patkos, A.C., Dimitris Plexousakis, and Yacine Amirat, *A Production Rule-Based Framework for Causal and Epistemic Reasoning.* 2012.
[41] Katerina Ksystra, N.T., and Petros Stefaneas, *On the Algebraic Semantics of Reactive Rules.* 2012.
[42] Gasperis, S.C.a.G.D., *Complex Reactivity with Preferences in Rule-Based Agents.* 2012.
[43] Iosif Viktoratos, A.T., and Nick Bassiliades, *Personalizing Location Information through Rule-Based Policies.*
[44] Lora Aroyo, G.-J.H., *Personalization on the Web of Data and New Paradigms for Distributed and Open User Modeling.* 2010.
[45] Masicampo, E.J., and Roy F. Baumeister, *Toward a physiology of dual-process reasoning and judgment: Lemonade, willpower, and expensive rule-based analysis.*, in *Psychological Science*. 2008.

APPENDIX I

Information representation in OWL format:

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
    xmlns="http://www.owl-ontologies.com/Ontology1435094805.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xml:base="http://www.owl-ontologies.com/Ontology1435094805.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="Message"/>
  <owl:Class rdf:ID="Work">
    <owl:disjointWith>
      <owl:Class rdf:ID="Other"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Family"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="Person"/>
        </owl:someValuesFrom>
```

```
      <owl:onProperty>
        <owl:TransitiveProperty rdf:ID="isWork"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Person"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Person">
  <owl:disjointWith>
    <owl:Class rdf:ID="Interests"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Context"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="MessageOrigin"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Family">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Email</rdfs:label>
  <owl:disjointWith>
    <owl:Class rdf:about="#Other"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Work"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#Person"/>
      <owl:onProperty>
        <owl:SymmetricProperty rdf:ID="isFamily"/>
      </owl:onProperty>
    </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:about="#Context">
  <owl:disjointWith>
    <owl:Class rdf:about="#MessageOrigin"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Person"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Interests"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Other">
  <rdfs:subClassOf rdf:resource="#Person"/>
  <owl:disjointWith rdf:resource="#Work"/>
  <owl:disjointWith rdf:resource="#Family"/>
</owl:Class>
<owl:Class rdf:about="#MessageOrigin">
  <owl:disjointWith rdf:resource="#Context"/>
  <owl:disjointWith rdf:resource="#Person"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Interests"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Interests">
  <owl:disjointWith rdf:resource="#MessageOrigin"/>
  <owl:disjointWith rdf:resource="#Person"/>
  <owl:disjointWith rdf:resource="#Context"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="DatatypeProperty_13">
  <rdfs:domain rdf:resource="#Family"/>
</owl:DatatypeProperty>
<owl:TransitiveProperty rdf:about="#isWork">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
```

```
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <owl:inverseOf rdf:resource="#isWork"/>
  </owl:TransitiveProperty>
  <owl:SymmetricProperty rdf:about="#isFamily">
    <owl:inverseOf rdf:resource="#isFamily"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:SymmetricProperty>
  <owl:FunctionalProperty rdf:ID="Influence">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="uplift">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:FunctionalProperty>
  <swrl:Imp rdf:ID="Rule-2">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >family</rdfs:comment>
  </swrl:Imp>
  <swrl:Imp rdf:ID="Rule-1"/>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.4.4, Build 579)  http://protege.stanford.edu -->
```