ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE ÉLECTRIQUE
M.Ing.

PAR
NESET SOZEN

STRUCTURE D'AGENTS EXPLORATEUR DE DONNÉES

MONTRÉAL, LE 22 DÉCEMBRE 2006

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

M. François Coallier, directeur de mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Oryal Tanir, codirecteur de mémoire
Bell Canada

M. Alain April, président du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC
LE 28 NOVEMBRE 2006
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

« STRUCTURE D'AGENTS EXPLORATEUR DE DONNÉES »

Neşet Sözen

SOMMAIRE

Beaucoup d'entreprises industrielles ont des bases de données ayant un contenu riche en données, mais pauvre en savoir. Néanmoins, elles contiennent toujours de l'information cachée qui peut être mise à jour et a une valeur marchande potentielle. L'exploration de données (data mining), aussi dit *fouille de données*, permet d'extraire du savoir à partir de grandes quantités de données en utilisant des méthodes tel que la classification automatique (clustering), apprentissage automatique (classification), les réseaux de neurones, etc.

Réaliser une application qui fait l'exploration de données d'une manière efficace requière des ressources hautement qualifiées et la mettre en opération dans une entreprise peut être très coûteux. Automatiser le processus d'exploration de données peut donc être une approche viable et abordable.

Une approche viable pour automatiser ce processus sera d'utiliser des agents, un type de programmes. Ces agents peuvent réaliser un ensemble de tâches de manière autonome et indépendante et vont collectivement travailler pour résoudre des problèmes complexes qui ne peuvent être résolus par des programmes monolithiques.

Ce projet consiste donc à réaliser un système d'exploration de données automatisé en utilisant des agents logiciels. Ce projet touche donc deux concepts fondamentaux : l'exploration de données et les agents logiciels.

L'exploration de données est une méthode itérative pour extraire des corrélations à priori inconnues à partir de grandes quantités de données, et ce, en utilisant des algorithmes dits d'intelligences artificielles ou provenant du domaine de la statistique.

Les algorithmes utilisés peuvent être catégorisés en deux groupes : les méthodes supervisées et non-supervisées. Les méthodes supervisées servent surtout à faire des prévisions futures à partir d'un modèle du système créé avec les données actuelles.

Avec ce type de méthode, il faut avoir des données étiquetées : entrée et sortie connue. Par exemple, dans une entreprise la fouille de données sera utilisée dans le contexte suivant : le lancement d'un nouveau produit sur le marché. Des algorithmes supervisés seront utilisés pour prédire les clients potentiels en utilisant des données reliées à un ancien produit similaire où chaque client est catégorisé comme bon client ou mauvais client pour ce produit. Les étiquettes sont « bon » et « mauvais ». Le modèle produit à

partir de ces données nous permettra de prédire la meilleure clientèle pour notre futur produit.

Les méthodes non-supervisées servent surtout à décrire le système, on veut donc comprendre le système. Avec ce type de méthode il n'y a pas de données étiquetées de départ, conséquemment aucune directive à priori sur ce que nous cherchons.

Le processus de fouille de données est composé des 5 étapes décrites ci-dessous :

1. *Définition du problème :* Cette étape consiste à établir l'énoncé du problème et les objectifs à atteindre. Pour se faire, les connaissances et l'expérience du domaine spécifique sont utilisées. Des hypothèses sont formulées pour décider comment mener le processus de « data mining ».

2. *Compréhension des données :* Durant cette étape, l'extraction des données et la détection de sous-ensemble de données « intéressantes » seront réalisées.

3. *Préparation des données :* Cette étape a pour objectif de préparer les données pour la prochaine étape. Typiquement, les données seront transformées pour avoir des résultats optimums dans l'étape suivante.

4. *Estimer un modèle :* À présent, un ensemble de méthodes et d'algorithmes sera utilisé pour créer des modèles à partir des données préalablement préparées. De plus, ces modèles seront vérifiés pour s'assurer que les objectifs fixés dans la première étape sont satisfaits.

5. *Interpréter un modèle :* Dans cette dernière étape, les résultats seront présentés au client dans une forme qui lui facilite la compréhension.

L'aspect itératif du processus fait que nous reviendrons à l'étape précédente tant et aussi longtemps qui nous ne sommes pas satisfaits des résultats de l'étape courante.

La méthodologie, employée pour créer notre système, s'est particulièrement inspirée de l'article [2]. Dans cet article, l'auteur propose des lignes directrices pour automatiser le processus de fouille de données. Globalement, ces directives peuvent se résumer ainsi : « L'automatisation du processus d'exploration de données est réalisée en se concentrant sur l'aspect opérationnel de ce processus et sur le domaine spécifique du client. ». De ce fait, le design du système d'explorateur de données était fait en deux phases. En premier lieu, toutes étapes du processus de « *data mining* » décrites ci-haut sont scrutées à la loupe pour proposer des solutions d'automatisation pour chacune des étapes. L'analyse était faite en considérant les objectives suivantes :

- Utiliser les informations à priori spécifiques au domaine du client pour automatiser le processus d'exploration de données

- Fixer les méthodes et les filtres, utilisés lors de l'estimation d'un modèle et lors des étapes de préparation de données respectivement.

Ensuite, la structure du système sera développée en considérant les points suivants :

- Les attributs de qualités principales sont la modifiabilité et la flexibilité.

- Design est basé sur JSR 73:JDM

- Les méthodes et les filtres utilisés pour l'exploration de données proviendront des librairies WEKA

- Le système est bâti sur la technologie des agents logiciels

    - Le développement et le design sont réalisés avec la méthodologie PASSI

    - Doit être conforme aux standards FIPA

    - Les agents seront créés en utilisant JADE

Le système d'exploration de données est destiné à extraire des connaissances à partir d'ODM « Operational Data Mart » de Bell Canada et aucune hypothèse à priori n'était proposée sur ce qui sera cherché par le système. De plus, il n'y avait aucune autre information que les données provenant d'ODM comme point de départ. Conséquemment, l'approche choisie pour mener l'exploration de données était de type descriptif en utilisant seulement des algorithmes non-supervisés. Une autre problématique à laquelle nous faisions face était la malédiction de la dimensionnalité causée par le grand nombre de dimension (nombre d'attributs utilisés pour représenter une donnée dans une base de données) des données d'entrée. La malédiction de la dimensionnalité est le fait que la demande en donnée augmente exponentiellement en fonction du nombre de dimension des données d'entrée. Comme démontré dans le document "*Use of unsupervised clustering algorithm in high dimensional dataset*", dans la section APPENDIX 2, les méthodes d'exploration de données classiques sont inefficaces dans notre cas et il fallait opter pour des méthodes non supervisées efficaces avec les données d'entrées de haut dimensionnalité d'attribut.

Ensuite, la mécanique opérationnelle de la deuxième étape de l'exploration de données a été inspectée. La « compréhension des données » est réalisée par un processus à quatre étapes :

- Identifier des attributs inappropriés et suspects
- Sélectionner la représentation d'attribut la plus appropriée
- Créer des attributs dérivés
- Choisir le sous-ensemble d'attributs optimaux

La troisième et la quatrième étape du processus de « data mining » étaient considérées de pair car « la préparation des données » et « estimer un modèle » sont relatives à un algorithme spécifique. La préparation des données consiste à choisir et à transformer les données pour l'algorithme d'exploration de données spécifique. Par exemple, pour l'algorithme PART, « la préparation des données » consiste à sélectionner tous les attributs de type numérique et à former un nouveau sous-ensemble de données qui sera utilisé pour créer un modèle lors de l'étape d'« estimer un modèle » car l'algorithme PART ne fonctionne qu'avec des données de type numérique. La quatrième étape comporte plusieurs tâches : bâtir un modèle, appliquer un modèle et tester un modèle. Notre système ne supporte que les deux premières tâches, car pour tester un modèle il faut avoir de l'information à priori (un modèle de référence avec des données d'entrée et les catégories ou classes correspondantes connues) dont nous ne possédons pas. Étant donné que notre système est basé sur JDM, nous nous sommes grandement inspirés du processus de « data mining » proposé par JDM pour établir les opérations à réaliser pour bâtir et appliquer un modèle.

Une des conditions que nous nous sommes fixée pour l'étape « estimer un modèle » était d'utiliser un ensemble d'algorithmes préétablis. Lors de notre sélection des méthodes, un ensemble de critères ont été adressés pour choisir une méthode de fouille de données destinée à être utilisée dans notre système :

- Exigences en connaissance à priori et la stratégie spécifique au domaine pour établir les exigences en connaissance à priori de la méthode d'exploration de données
- Les méthodes non supervisées seulement
- Sensibilité de la méthode à la haute dimensionnalité des données d'entrée.
- Extensibilité des méthodes. (c.-à-d. introduire du parallélisme, étendre les calculs sur plusieurs hôtes)
- Complexité en temps d'exécution

Notre architecture est façonnée par deux technologies (ou concepts) : les agents logiciels et JDM. Généralement, tout système utilisant des agents est composé de quatre composantes abstraites : la couche plateforme « platform layer », le système de gestion d'agents « agent management system », l'architecture agent « agent architecture » et la couche domaine « domain layer ».

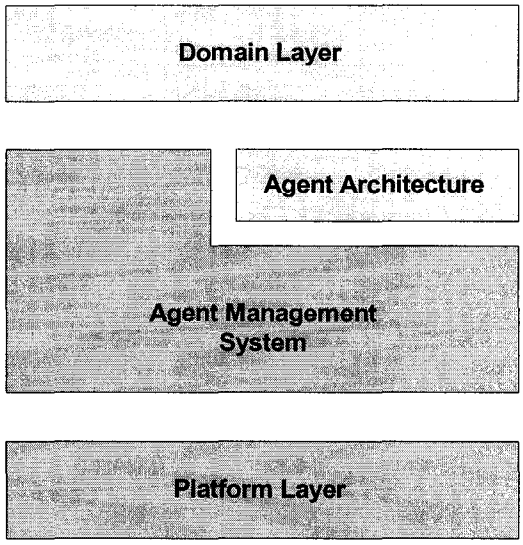


Figure 1 Architecture générale d'application basée sur les agents logiciels.

La couche plateforme correspond à l'hôte sur lequel le système sera exécuté. Le système de gestion d'agent fournit un environnement aux agents pour accéder à la plateforme et exister. L'architecture agent représente nos agents. La couche domaine relate des aspects spécifiques au domaine.

JDM propose une architecture avec trois composantes logiques : l'interface de programmation d'application (Application Programming Interface "API"), le moteur de « data mining » (Data Mining Engine "DME") et le dépôt d'objet de « data mining » (Mining Object Repository "MOR"). API est une abstraction permettant d'accéder aux services fournis par DME. DME encapsule tous les services de « data mining ». MOR contient tous objets d'exploration de données (les modèles, des statistiques, etc.) produit par DME.

L'architecture résultante est montrée dans Figure 2.

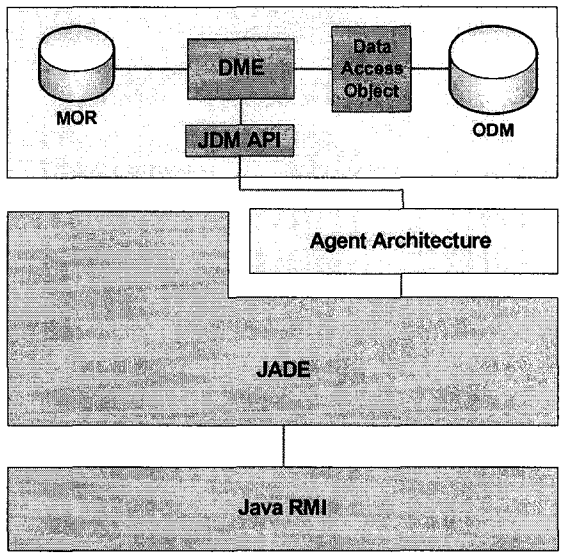


Figure 2 Architecture du système explorateur de données automatisé

Le système fonctionne sur la plateforme Java RMI. Nos agents sont implémentés avec la librairie de JADE, conséquemment ce dernier sera sélectionné comme système de gestion des agents. Étant donné que le domaine spécifique de notre projet est le « data mining », la couche domaine est créée en se basant sur l'architecture proposée par JDM.

Lors de nos expérimentations, nous nous sommes surtout concentrés sur l'impact de la haute dimensionnalité des attributs des données sur les résultats. Pour se faire, trois méthodes ont été choisies. K-means qui est une méthode de classification non-supervisée classiques, fuzzy c-means qui est aussi une méthode de classification non-supervisée dont chaque point appartient à une classe à un certain degré (à un certain pourcentage) et PART « projective adaptive resonance theory » est un algorithme de réseaux de neurones adapté pour la recherche dans les sous-espaces d'attribut. Nous avons effectivement observé que la dimensionnalité des données d'entrée avait un impact majeur sur la qualité des résultats, de plus la majorité des algorithmes classiques ne sont pas adaptés pour extraire de l'information à partir d'un ensemble de données de haute dimensionnalité.

En conclusion, dans ce projet, toutes les façades du processus d'exploration de données sont explorées dans le but de l'automatiser en utilisant des agents autonomes. Lors de notre investigation, il a été remarqué que les étapes de compréhension des données et préparation des données consomment beaucoup de temps et elles ont un impact majeur sur la qualité des résultats de recherche. De plus, lorsque des méthodes complexes et hautement spécialisées qui sont utilisées, généralement leurs paramètres nécessitent beaucoup d'ajustement et ils ne marchent qu'avec un type de donnée spécifique

seulement. Par exemple, PART ne peut être appliqué que sur des nombres naturels (c-à-d N={0,1,2,3,...}) et les valeurs choisies pour les deux paramètres d'entrée qui sont « paramètre de vigilance » et « paramètre de vigilance de distance » ont un impact sur la grandeur (nombre d'attribut qui définisse le groupement) des groupements d'objet qui peuvent être trouvés. Donc, il sera plus rentable de se concentrer sur ces deux étapes autant (même plus) que sur les algorithmes de recherche utilisés lors de l'estimation de modèle. En ce qui concerne les algorithmes à utiliser pour faire l'exploration de données, nous faisions face à la malédiction de la dimensionnalité causée par le nombre élevé que possédaient les données d'entrée. Pour résoudre ce problème nous ne pouvions utiliser les solutions classiques comme la réduction du nombre de dimension des données d'origine car, ces méthodes de réduction nécessitent des informations à priori sur les données d'origine. La seule option restant était d'utiliser des algorithmes spécialement adaptés à des problèmes comportant des données de hautes dimensionnalités.

À la suite de notre analyse du processus de « data mining », nous avons débuté le design du système basé sur les agents. Compte tenu de la complexité inhérente des systèmes basés sur les agents, un ensemble d'outils a été sélectionné pour simplifier le design et le développement du système. La technologie des agents logiciels est un domaine très nouveau, conséquemment il est très difficile de trouver les outils adéquats. À la suite d'une recherche intensive et quelques essaies et erreurs, la méthodologie PASSI fut choisie pour le design et le développement du système. PASSI est une méthode itérative de design et développement de système multi-agents facile à utiliser et à comprendre. Sa facilité est due à utilisation de la notation UML et le « toolkit » PTK qui est une extension de Rational Rose qui implémente PASSI, ainsi qu'une panoplie de documents et d'articles sur cette méthode. De la même manière, nous avons choisi la plateforme JADE pour implémenter nos agents.

Compte tenu de la grande envergure du projet, l'objectif principal de ce mémoire était de faire un travail préparatoire pour des projets de recherches futures dans le domaine. Plus d'analyse et d'information à priori sont nécessaires pour avoir un système optimal. En ce moment, les filtres utilisés lors de la préparation des données sont plutôt élémentaires. Il faut aussi étudier la possibilité d'utiliser les algorithmes de classification hiérarchique (hierarchical clustering), les méthodes d'exploration de données temporelles, les arbres de recherche, ainsi que l'exploration de données en parallèle et distribuée. Il faut aussi étudier les agents mobiles, l'impact de leur utilisation dans notre contexte. Les agents mobiles sont des agents logiciels qui peuvent se déplacer d'un hôte à un autre.

DATA CRAWLER AGENTS FRAMEWORK

Neşet Sözen

ABSTRACT

This document presents the framework of agent-based automated data mining system searching for knowledge in data with high dimensional feature space using unsupervised methods. First, the data mining process was analyzed to establish all the operational mechanics of the process. The impact of the high dimensionality of the data on unsupervised methods (i.e. clustering methods) was also studied. Then, the agent-based system was designed and developed with PASSI methodology and JADE platform.

ACKNOWLEDGEMENT

I am grateful to Department of Software and IT Engineering, École de technologie supérieure, where I did the entire research on data crawler agents framework and specially to M. François Coallier and M. Oryal Tanır for their support and assistance during this project.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF ALGORITHMS

Page

ACRONYMS

DM	Data Mining
KD	Knowledge Discovery
DDM	Distributed Data Mining
DMT	Data Mining Techniques
DCS	Data Crawler System
ODM	Operational Data Mart of Bell Canada
PRA	Perceive-Reason-Act
WEKA	Waikato Environment for Knowledge Analysis
JDM	Java Data Mining
JADE	Java Agent Development Framework
PADMA	Parallel Data Mining Agents
BODHI	Beseizing Knowledge trough Distributed Heterogeneous Induction
JAM	Java Agents for Meta-Learning
PPML	Predictive Model Markup Language
KIF	Knowledge Interchange Format
RDF	Resource Description Framework
RUP	Rational Unified Process
OO	Object oriented
AO	Agent oriented
PASSI	Process for Agent Societies Specification and Implementation
PTK	PASSI tool kit
UML	Unified Model Language
DAO	Data Access Object

INTRODUCTION

Most Industrial company databases are rich in data but weak in knowledge. On the other hand there is always a lot of hidden knowledge to be extracted. Potentially, the hidden information and knowledge can be converted to an opportunity resulting in major revenues. *Knowledge Discovery* (KD) is a process aimed at extraction of previously unknown and implicit knowledge from large databases, which may potentially be of added value for some given application [1]. KD process is accomplished by *Data Mining* (DM) process using *Data Mining Techniques* (DMT). There are many DMT like classification, clustering, etc. These techniques are described in more detail in further paragraphs.

Implementing the data analysis technology and use of DMT efficiently requires highly qualified resources and can be very costly to put into operation "in-house" data mining systems or to subcontract the project to a third-party. Most of the time, the organizations don't have the resources available to afford these options. To automate the KD process could be viable at a reduced cost [2].

A viable option for automating KD process, considering that in general during this process we need to scale up with massive data sets, is agent-based applications. An agent-based (or multi-agent) system is a system made up of several agents, working collectively to resolve complex problem that will be difficult to achieve by a single agent or system. An agent is a software program that realizes a set of tasks and has its own execution thread.

This research project is about implementing an automated DM system using agents. In this paper, we briefly review existing DM systems and frameworks and identify the challenges to overcome in order to automate the DM process and present the architecture of our system.

This paper is complemented by two other papers: "*Data Crawler System Requirements and Specifications*" in APPENDIX 1, which contains the software requirements and design details of the Data Crawler System (DCS) and "*Use of clustering algorithms for knowledge extraction from high dimensional dataset*" in APPENDIX 2, which describes an analysis made with several clustering methods in context of high dimensional data where a strategy for selecting data mining methods is established.

1.1 Data Mining Concepts

Since our research project is about automating the data mining process, in this section the data mining process itself is described in detail and the strategy to automate this process is exposed.

1.1.1 What is Data Mining?

Data mining is an iterative process for discovering unknown knowledge from large volumes of data by applying statistical and machine learning techniques. At a high level, knowledge discovery and data mining process can be seen as follow:

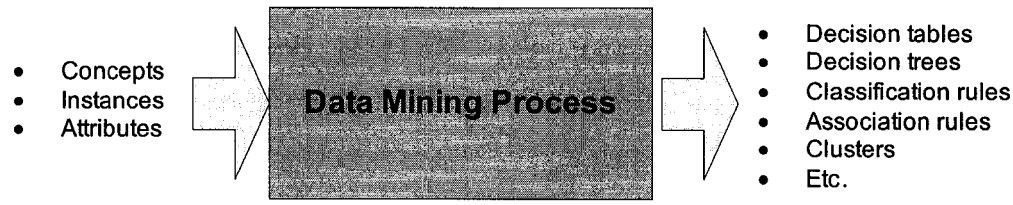


Figure 3 Data mining process's input and output

As shown in Figure 3, the inputs to the data mining process are instances, attributes and concepts [8]. The instances are things that the DMTs will be applied on and the

attributes are characterization of each instance. For example, each row of the database in Figure 4 corresponds to an instance and each column corresponds to an attribute.

ID	Age	Sex	Movie Type	Occupation
1	32	F	Suspense	Doctor
2	23	F	Adventure	Plumber
3	21	M	Action	Taxi driver
...	...	...	...	...

Figure 4 A database illustration

The concept [8] is what will be learned. For example, if the DMT is classification, the expected outcome will be a set of classified instances and if the technique is decision tree, the outcome will be association rules between attributes and not a class. In spite of the learning scheme, what is learned is the concept.

These outcomes are *knowledge*. The knowledge is structural patterns in data, discovered by machine learning methods (i.e. DMTs). The knowledge will have different representation depending on the used DMT. If the used technique is a decision tree construction algorithm the knowledge will be in the form of a decision tree and in case of clustering methods, the knowledge will be represented by clusters. The concept of knowledge and knowledge representation will be detailed in CHAPTER 3.

Descriptive DM vs. Predictive DM

There are two approaches (or objectives) for using data mining: *prediction* and *description*. Prediction is about forecasting future data values. This type of data mining will produce a model of the system based on the given data. The descriptive data mining will produce hidden knowledge patterns without a preset hypothesis about what the outcome may be. The goal is to gain an understanding of the system.

With a predictive approach, there is a question to answer, for example "what might be a good promotion for our new product?", "How much profit can be made for the next quarter?". With descriptive approach there is a valuable data with no prior directive for what we are looking for.

Predictive approach is the most commonly used approach in the industry and also the easier one because we already have some comprehension of the input data. For example, the distribution of the data is known (normal distribution, Poisson distribution, etc.), the number and the type of classes are known and there is a data set with target output (i.e. classes, clusters, etc.) and so on. Therefore, we can predict the future using a model produced with a given data set that has a priori information.

On the other hand, the descriptive approach is rarely used in the industry because it is very complex to conduct and we can never be sure of the validity of the produced models. To begin with, we don't have any point of reference (i.e. a data set with target output) to compare or to verify the quality of the produced models. Therefore, we can never say if the produced model is poor or good. Also, the descriptive DMTs are very complex to execute properly since it's difficult to select the right values for the input setting parameters for our domain of problem and mostly the DMTs are highly specialized for a specific context or a specific type of data, which aren't necessarily fully compliant with our domain of problem.

Data Mining Techniques

DMTs are algorithms, machine learning methods or functions used to extract knowledge. There are many DMTs, each of which falls into one of these following categories [3]:

1. *Classification* – discovery of a predictive learning function that classifies a data item into one of several predefined classes.

2. *Regression* – discovery of a predictive learning function, which maps a data item to a real-value prediction variable.

3. *Clustering* – a common descriptive task in which one seeks to identify a finite set of categories or clusters to describe the data.

4. *Summarization* – an additional descriptive task that involves methods for finding a compact description for a set (or subset) of data.

5. *Dependency Modeling* – finding a local model that describes significant dependencies between variables or between the values of a feature in a data set or in a part of a data set.

6. *Change and Deviation Detection* – discovering the most significant changes in the data set.

The DM techniques that will be used in our system will be discussed in section 5.4.

Data Mining Process

The steps of the knowledge discovery and data mining process, shown in Figure 5, are described below.

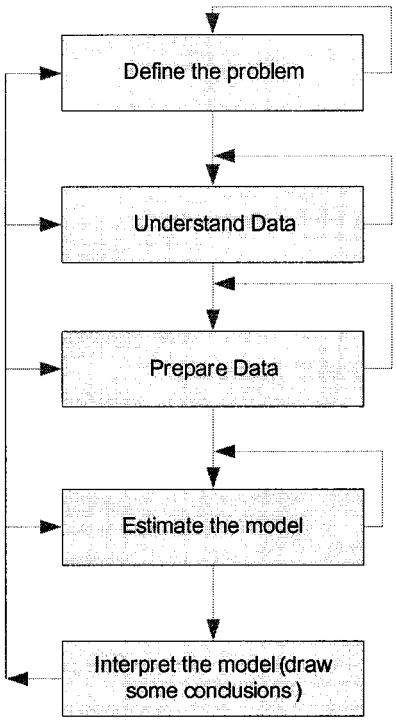


Figure 5 Data mining process

1. *Define the problem:* In this initial step a meaningful problem statement and the objectives of the project are established. Domain-specific knowledge and experience are usually necessary. In this step, a modeler usually specifies a set of variables for the unknown dependency and, if possible, a general form of this dependency as an initial hypothesis. There may be several hypotheses formulated for a single problem at this stage. The first step requires the combined expertise of an application domain and a data-mining model. [3]

2. *Understand Data:* This step is about data extraction and detection of "interesting" data subsets.

3. *Prepare data:* During this step the final dataset is constructed. Common data preparation tasks will be outlier detection and removal, scaling, encoding,

selecting features, etc. This step should be regarded as together with other data mining steps. Hence, a good data preparation method with a priori knowledge will provide optimal results from a data mining techniques.

4. *Estimate the model:* In this step, various data mining techniques and algorithms are applied on data set prepared previously. And the verification of the DM models to ensure that our model is robust and achieve the objectives specified during the problem definition.

5. *Interpret the model:* In this final step, the results are presented to client (decision maker). Since, the results are used to make decision; they should be understandable in form of simple reports using organization templates and not hundreds of meaningless numerical results.

Even most of the DM tasks (e.g. prepare data, building model, applying model, etc.) are accomplished during step 3 and 4, a good understanding of the whole process is important for any successful application [3].

1.1.2 How to automate data mining process?

Automating the data mining process is achieved by focusing on the operational aspects of the data mining process and specific client domains [2].

Since, all data-based systems are designed within a particular application domain. Domain-specific knowledge and a priori information are important to perform a successful automated data mining system. Therefore, a priori information can be used to automate each step of the data mining process in Figure 5. Therefore, by focusing on client domains data understanding phase of DM process is performed with a minimal human intervention and offering a fixed set of data mining technique and analysis

solutions allow templatizing the problem definition and deliverables. Moreover, focusing on a particular problem domain gains the domain knowledge by increasing the probability of successful solution delivery in the future.

1.2 Agents Theories

The agent field is very new and there are many definitions and concepts about it. The definition of agents and related concepts used in this research project are based on [11].

1.2.1 What is an agent?

An agent is a software system that is situated in an environment and that operates in a continuous Perceive-Reason-Act (PRA) cycle as illustrated in Figure 6.

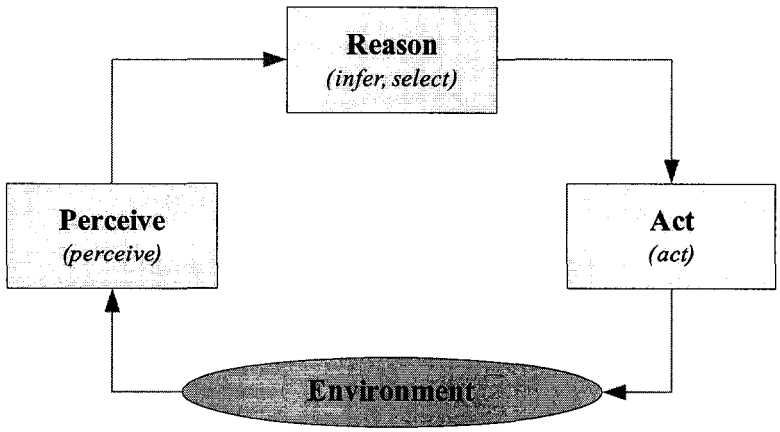


Figure 6 Perceive-Reason-Act Cycle

Accordingly, the agent receive some stimulus from the environment and this stimulus is processed within the perceive component. Then, this newly acquired information is combined with the existing knowledge and goals of the agent by the reasoning component. Then, this component determines possible actions of the agent and the best actions are selected and executed by the act component.

To be more concise and formal the agent is represented by the following 7-tuple where S represents the environment:

$$agent = \langle D, T, A, perceive, infer, select, act \rangle$$

where,

D = database that contain the agent's acquired knowledge

T = a set of partitions of the environment S

A = a set of possible actions of the agents

$perceive : S \rightarrow T$

$infer : D \times T \rightarrow D$

$select : D \times T \rightarrow A$

$act : A \times S \rightarrow S$

Figure 7 explains the information flow between each component of the agent.

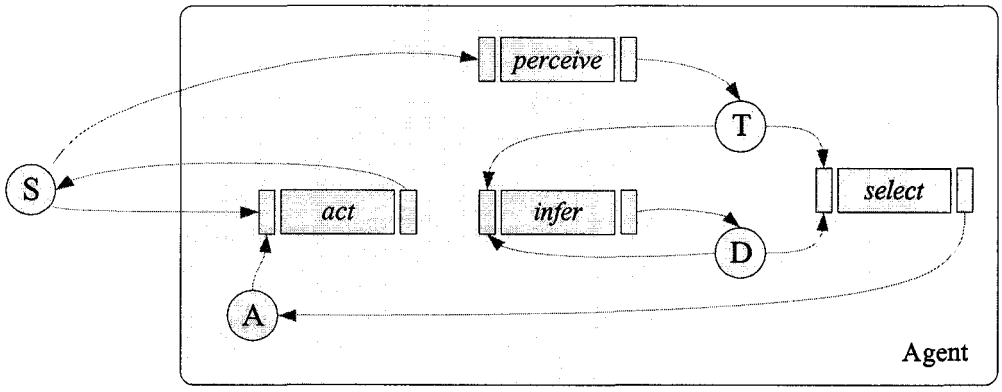


Figure 7 Agent Formal Representation

Role

Role has several definitions and we will stick to the following "A role is the functional or social part which an agent, embedded in a multi-agent environment, plays in a (joint)

process like problem solving, planning or learning" [12]. The major characteristics of a role are the following:

- There exist mutual dependencies between roles. Some roles can only exist if other roles do exist. For example, the role of a "teacher" only makes sense if the corresponding role of (at least one) "student" exists as well.

- A member of a society can play several roles even at the same time. This property is called role multiplicity and can lead to so-called role conflicts.

According to definitions above an agent can play several roles from a set of roles R. For example, a person can be an engineer and a teacher at same time. Our previous 7-tuple description of agent will be redefined according to concept of role as follow:

$$agent = \langle D \cup D_r, T, A \cup A_r, perceive \cup perceive_r, infer \cup infer_r, select \cup select_r, act \cup act_r \rangle \quad \text{with } r \in R$$

General Architecture

Agent architecture is defined as follow

> *Agent architecture is a structural model of the components that constitute an agent as well as the interconnections of these components together with a computational model that implements the basic capabilities of the agent.*

The selection of a particular type of the architecture instead of another will have a huge impact on the system. The following relation portrays the relation between agent, role and architecture

$$agent = roles + architecture$$

This relation, according to the definitions above can be illustrated by a conceptual representation as in Figure 8.

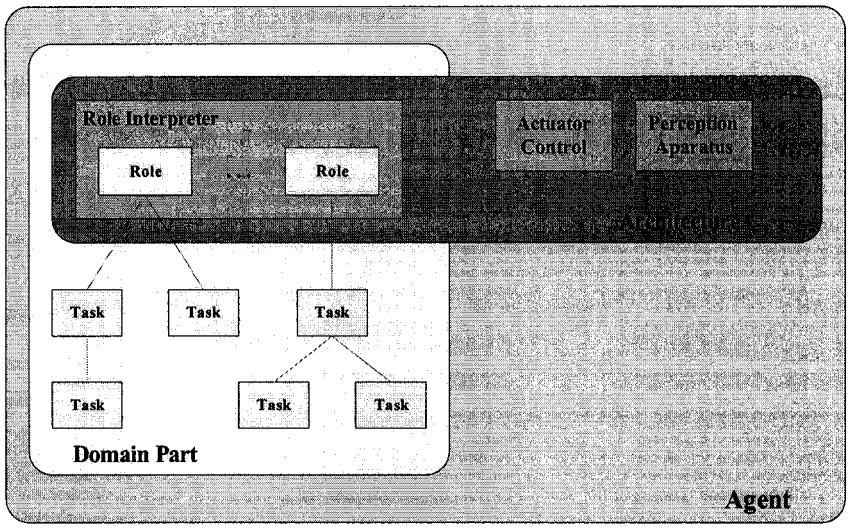


Figure 8 Agents, Roles and Architecture

The architecture, enclosed in agent concept, contains perception and actuation subsystem as well as the role interpreter. The architecture is domain independent. The role interpreter links the architecture to the domain specific aspects of different roles represented by a task tree.

The conceptual representation in Figure 8 will determines the basis structure of agent-based applications. The generic application architecture is depicted in Figure 9.

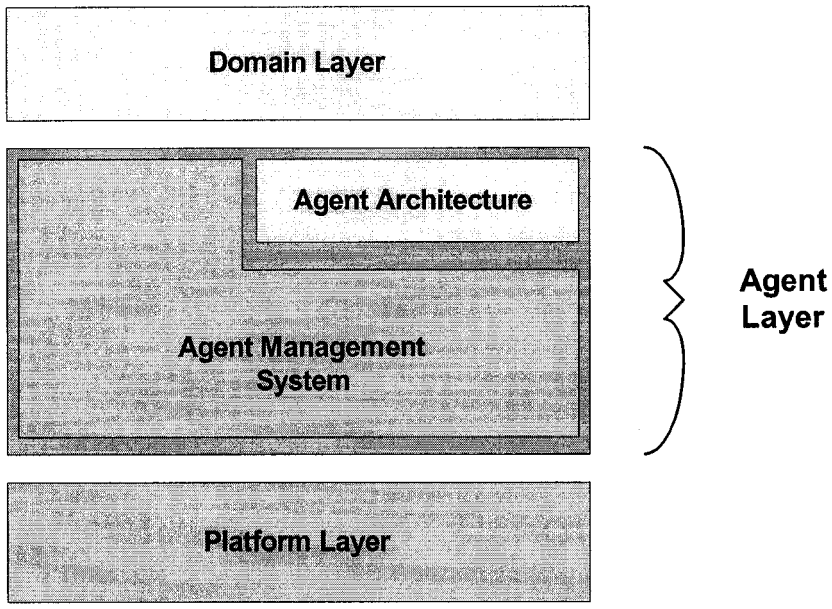


Figure 9 Generic agent-based application architecture

The architecture of a generic agent-based application is formed by three layers: platform layer, agent layer and domain layer.

Platform Layer

It is the basis layer that hosts the application. In case of more complicated applications, it may be spread over several platforms.

Agent Layer

This layer contains two major elements. The Agent Management System provides the interface between the agent architecture and the hardware platform. It provides all elements necessary for agents to exist and live. The Agent Architecture represents the domain dependent roles of the agent.

Domain Layer

This layer implements the domain specific aspects of the system.

Systems of Agents

We have a multiagent system when several intelligent agents exist within the same environment. The environment can be different depending on the agents. When we have robotic agents it will be a physical environment, in case of software agents it will be a runtime environment or virtual reality. A multiagent system can be formalized as follow

$$\left\{S, \left\langle D, T, A, perceive, infer, select, act \right\rangle_i\right\}$$

where S is the environment and each 7-tuple represents a specific agent identified by *i*. The main feature of multiagent systems is that a major part of the systems functionality is not explicitly and globally specified, but it emerges from the interaction between agents that constitute the system. Therefore, interaction is the main aspect of multiagent systems. Interaction is defined as follow:

> *Interaction is the mutual adaptation of the behavior of agents while preserving individual constraints.*

This general definition of interaction is not limited to explicit communication or message exchange as a predominant means in the multiagent literature. It will focus on mutual adaptation, which means that the participating agents co-ordinate their behavior. Also, the interaction definition will focus on balancing between social behavior manifested in the mutual adaptation and the self-interest of agent. To get the best global results from the system, it is important to have agents within a multiagent system with a mix of self-interest and social regard that will value the performance of the entire society and their individual performance.

Thus, social dimension is important in agent-based systems. The social structure of a society determines how the entities within the society relate to each other. Before going

any further, we will use some definitions taken from sociology and organizational theory to explain agent societies. Structure, society and social system are defined as follow:

A structure is a collection of entities that are connected in a non-random manner

A society is a structured set of agents that agree on a minimal set of acceptable behaviors

A social system is a society that implements a closed functional context with respect to a common goal.

The definitions above are used to achieve a social dimension of an agent-based system. The structural connections between agents in an agent society are shown in Figure 10 where each agent interacts with each other.

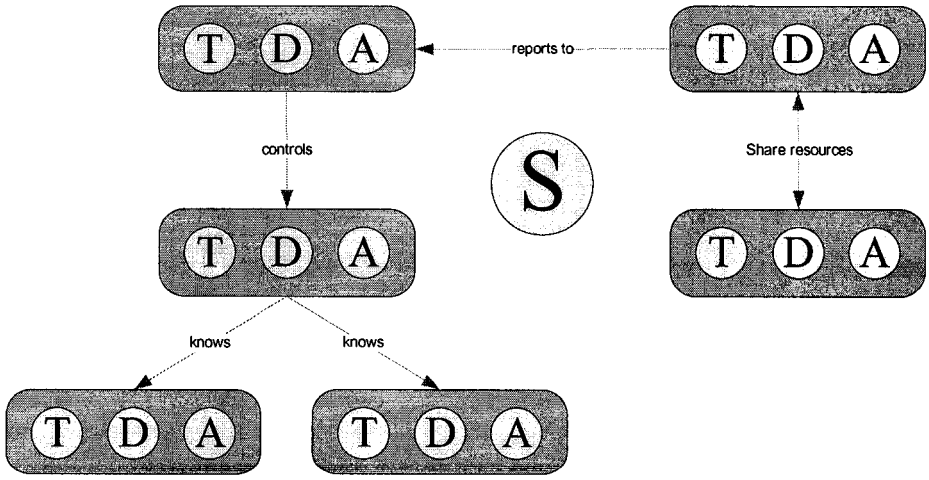


Figure 10 Agent society diagram

1.3 Agents in data mining: Issues and benefits

Agent-based systems require a platform for agents to function and accomplish their tasks. It will increase the complexity of the whole system. Therefore, we should ask our selves if agents are the right technology for a given problem.

In knowledge discovery, we are trying to solve very complex problems where the boundary of the problem domain isn't properly defined (or not defined at all). Mostly, the data mining techniques are used to define those boundaries by extracting meaningful correlation within data. And, the key issues of an agent-based system are the permission of incomplete knowledge about the problem domain and the possibility to build an intelligent and dynamic system [15]. Therefore, the system can reason, perform tasks in a goal driven way and react to a changing environment. Another advantage will be the possible solution to the problem using distributed systems since each agent encapsulates the behavior and the state. All these advantages are detailed in the following subsections.

Scalability of DM and dynamic data gathering

Our data mining system will be processing data from Bell ODM, which is a huge data repository. Also several thousands of operational data is inputted to this database every day. Processing all data from this data mart at once is practically impossible and scaling up to this massive data set is a necessity. Therefore, we need to apportion the data between agents. The use of agents will facilitate dynamic selection of sources and data gathering. DM system will select and process data autonomously, without human intervention. Also, we can scale up (or scale down) number of agents processing data if required in future which is more difficult to accomplish with a static system without agents.

Multi-strategy DM

As mentioned, we are doing descriptive data mining. A multi-strategy DM will be more suitable as we need to apply multiple data mining techniques and in some case combination of several techniques. As mentioned in [13], appropriate combination of multiple data mining techniques may be more beneficial than applying just a particular one. DM agents may learn in due course of their deliberative actions which one to choose depending on the type of data retrieved and mining tasks to be pursued. Multi-strategy DM is an argument in favor of using agents, as mentioned in [13].

Interactive DM

According to [13], pro-actively assisting agents drastically limits the amount a human user has to supervise and interfere with the running data mining process, e.g., DM agents may anticipate the individual limits of the potentially large search space and proper intermediate results.

In [13], the authors treat the role of agents in distributed data mining (DDM). With DDM, the data, instead of been centralized in a single repository, will be distributed over several data source. Therefore, besides the issues of scalability, DDM is also challenged with autonomy and privacy. The privacy is about the ensuring data security and protecting sensitive information against invasion by outsiders, which is not a concern for us because our data crawler system will be local and it will access directly a single data repository which is Bell's ODM. Autonomy is an issue when the DM agents are considered as a modular extension of a data management system to deliberatively handle the access to the data source in accordance with constraints on the required autonomy of the system, data and model. Once again, autonomy too isn't a concern because our system is designed in accordance with Bell's operational data. Our DM system will fit

the client's data because in order to have an automated DM system we need to focus on specific clients domains [2].

1.4 Objectives

In this section, the objectives of the system will be resumed. The main objective of this research project is knowledge extraction from data coming from Operational Data Mart (ODM) which is a large data repository administered and used by the Bell Business Intelligence and Simulation team. The knowledge discovery should be accomplished autonomously. There is no prior hypothesis on exploration, thus the selected approach is descriptive data mining.

A secondary objective is to automate the data mining process using available domain specific a priori information. Fixed sets of DM methods will be used for modeling phase and filters for the data preprocessing phase. The designed system will be implemented using the Java language and WEKA libraries for data mining methods and data preparation filters. The agents that will perform the data mining process will be implemented using JADE (Java Agent DEvelopment Framework). More details on designed system are revealed in CHAPTER 8.

In our research project, computing time and responsiveness of the system is not an issue since we are working on a data mart; the designed system will not be online. Here online doesn't mean that the system is connected to the net. It means that the system needs to classify quickly in order to accomplish the right action. (e.g. systems that classify the fruits in different boxes according to the classification result). Performance will be considered during the design but it will not be the primary attribute of our system.

Flexibility and modifiability are primary attributes of our system considering that the data crawler system is an academic project that will evolve and be improved eventually.

New components will be added or existing components will get changed. As we select descriptive DM approach, the DMTs will be changed or adapted in order to perk up the quality of the produced knowledge.

DM methods adapted to our client's domain should be selected. Once again because of the mined data characteristics and the selected data mining strategy (i.e. descriptive approach) compelled by the lack of a priori information, in this project only unsupervised DMTs that are proficient in high dimensionality can be used.

Our system should be compliant with FIPA standards.

CHAPTER 1

STATE OF THE ART

The common practice in this field is the automating of knowledge discovery and data mining process by means of autonomous data crawlers or software agents. The most prominent of agent-based DM systems are: PADMA [17], BODHI [18], Papyrus [19] and JAM [20].

PADMA (PArallel Data Mining Agents), as its name suggests, is an agent-based parallel data mining system where each agent performs data analysis locally. Then the local models are collected and used in a second higher-level analysis to produce a global model. The hierarchical clustering is used by the agents to classify unstructured text document and to visualize web based information. The architecture of the system is based on this clustering algorithm.

BODHI (Beseizing knOwledge through Distributed Heterogeneous Induction) System is a distributed knowledge discovery system that exists to be used in context of collective data mining within heterogeneous data sites. The system is implemented with Java to prevent it being bound to a specific platform. It offers runtime environments (agent stations where agents can live) and a message exchange system that supports mobile agents. The data mining process is spread out across local agent stations and agents moving between them. Each mobile agent will hold its state, data and knowledge. A special purpose agent "facilitator" is responsible for initializing and coordinating the DM tasks performed by the agents. The communication and the control flow between agents also are coordinated by the facilitator agent.

JAM (Java Agents for Meta-Learning) is an agent-based system that use meta-learning to do DM. The meta-learning is a technique that seeks to compute a global classifier

from large and inherently distributed databases. Several learning algorithms such as ID3, CART [32], BAYES [31] and WPEBLS [33] can be applied on heterogeneous databases by a learning agent that may be locally stored on a site (Datasite) or imported from other peer sites that compose the system. JAM is a network of Datasites where JAM agents and other objects (local database, learning and meta-learning agents, GUI interface, etc.) reside on. Each Datasite's learning agent builds classification models using a different technique and the meta-learning agents builds meta-classifiers by combining multiple models learned at different sites. Once the combination of classifiers is achieved, the JAM system manages the execution of these modules to classify data sets of interest located in any Datasite.

Papyrus is a designed over data clusters and meta-clusters. It supports predictive model strategies including C4.5. Papyrus focus on doing data intensive computing locally. Hence, mobile agents move data, intermediate results and models between clusters to perform computation locally. Each cluster is represented by one distinctive node that is an access and control point for the agents. The coordination of the overall clustering task is achieved by a central root site or dispersed to the network of cluster access points. PMML (Predictive Model Markup Language) is used by Papyrus to describe predictive models and metadata, which facilitate their distribution and exchange.

All those agent-based systems represent a potential solution to our problem and they all have advantages and disadvantages, which make them suitable or unsuitable to our case as shown in Table I.

Table I

Several agent-based DM systems comparison

System name	Advantages/Disadvantages	Why not used
PADMA	*Advantage:* • Hierarchical clustering algorithm *Disadvantages:* • Text mining only • Only hierarchical clustering algorithm	Even if PADMA is based on an unsupervised algorithm: hierarchical clustering algorithm and it can only use this algorithm which is very limitative. Also, it only does text mining.
BODHI	*Advantages:* • Distributed knowledge discovery • Collective data mining • Good architecture • Java *Disadvantages:* • Supervised learning • Several heterogeneous data sites (different from our context: one central data site)	BODHI propose a very interesting architecture and collective data mining is a very interesting concept but it is based on supervised learning, which is in contradiction with our descriptive approach (i.e. unsupervised learning). Also, the fact of several heterogeneous data sites is very different from our context where we have one huge data site.
JAM	*Advantages:* • Meta-learning • Several DMTs *Disadvantages:* • Only supervised DMTs • Distributed data sites	Use only supervised DMTs.
Papyrus	*Advantages:* • PMML • Meta-clusters *Disadvantage:* • Predictive models only	Use only supervised DMTs..

First, in our research project a descriptive data mining strategy was selected and most systems, in Table I, are designed for predictive data mining. Secondly, their design focus on distributed data mining (DDM) and trying to resolve problems related to the aspect of dispersion data through heterogeneous data sites which wasn't our case since all input data was located in a central database (ODM).

Our research project is mainly influenced by the article "Data Mining as an Automated Service" [2], which proposes some strategies to automate the knowledge discovery and data mining process. The author in this document considers the data mining process as a service and the organization that want to use the KD as a client. The high-level goals of the automated data mining services are determined as

1. Providing the client with a high quality, pertinent results.

2. Removing or minimizing the amount of human intervention into DM process to produce high quality, pertinent results.

These goals are achieved by focusing on the problem domains (market verticals). In [2], the required processes, issues and challenges in automating data mining are described. The approach proposed by the author can be applied to resolve our problem.

Also, in "Automating Exploratory Data Analysis for Efficient Data Mining" [4] the authors propose a number of approaches to automate the data understanding and data preparation steps. In this document, some strategies are proposed for optimizing the dataset before mining them with DMTs. Detecting inappropriate and suspicious attributes, target dependency analysis, creating derived attributes are some of these strategies. These approaches can be used in our research project too but they need to be adapted to our context.

Agent-based system design and development

An agent-based system can be developed using common development methodology such as RUP. Considering the complexity of the agent-based system, using agent-oriented methodologies will make the development more straightforward.

Since agents are still a forefront issue, there are many methodologies proposed and all of them have interesting features and capabilities. Most of them are still in beta version and don't have a widespread acceptance in the agent community. The following diagram shows all existing agent-oriented methodologies and their relations to each other.

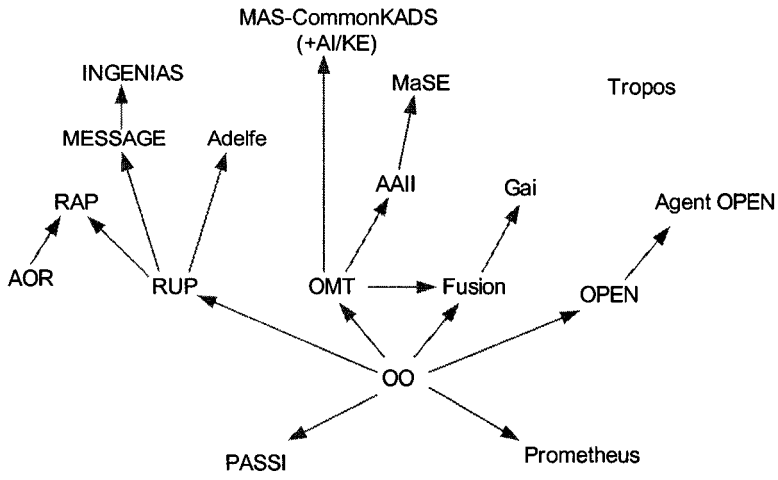


Figure 11 Genealogy of Agent-Oriented Methodologies [21]

Our goal in this project isn't to test and review all existing agent-oriented (AO) methodologies but to use one of them. After, testing few of them the PASSI methodology was selected, because it was easily applicable and implemented in a toolkit PTK which allows the design of an agent-based system using Rational Rose with the methodology. Further details on PASSI methodology are found in section 2.1.

CHAPTER 2

METHODOLOGY

In this section the research project methodology used for the creation of the Data Crawler System framework is described.

We propose to follow each of the five steps of the data mining process presented in Figure 5. We also propose to automate each of them. The approaches to automate the data understanding and data preparation steps of DM process found their basis in the work presented in [4]. The proposed solutions in [4] are not used as in the document since we needed to adopt them to our problem domain. Next, the model estimation step is detailed as well as the selected data mining algorithms and how they were selected. Finally, the model interpretation step is described. The following objectives are reflected in our analysis to automate the data mining process:

- Automate the data mining process using domain specific a priori information
- Fix the set of DM techniques that will be used for the modeling phase and filters for the data preprocessing phase

Once the data mining process is analyzed in order to automate it, the data mining system is designed. The following objectives are reflected in the data mining system design

- The designed system will be implemented using the Java language
- Design is based on (or inspired by) JSR 73:JDM
- WEKA libraries for data mining methods and data preparation filters are used
- Primary attributes are flexibility and modifiability
- Use of agents technology to build and design the system

- Use a methodology specialized for developing agent-based system to facilitate the design/development process
- Must be in compliance with FIPA standard
- Use of JADE software framework to construct agents

2.1 PASSI Methodology Description

The Data Crawler System is designed and developed following the PASSI methodology.

PASSI [40] (Process for Agent Societies Specification and Implementation) is a methodology for designing and developing multi-agent systems, using a step-by-step requirement-to-code process. It integrates the design models and concepts, using the UML notation, from two dominant approaches in the agent community: OO software engineering and artificial intelligence.

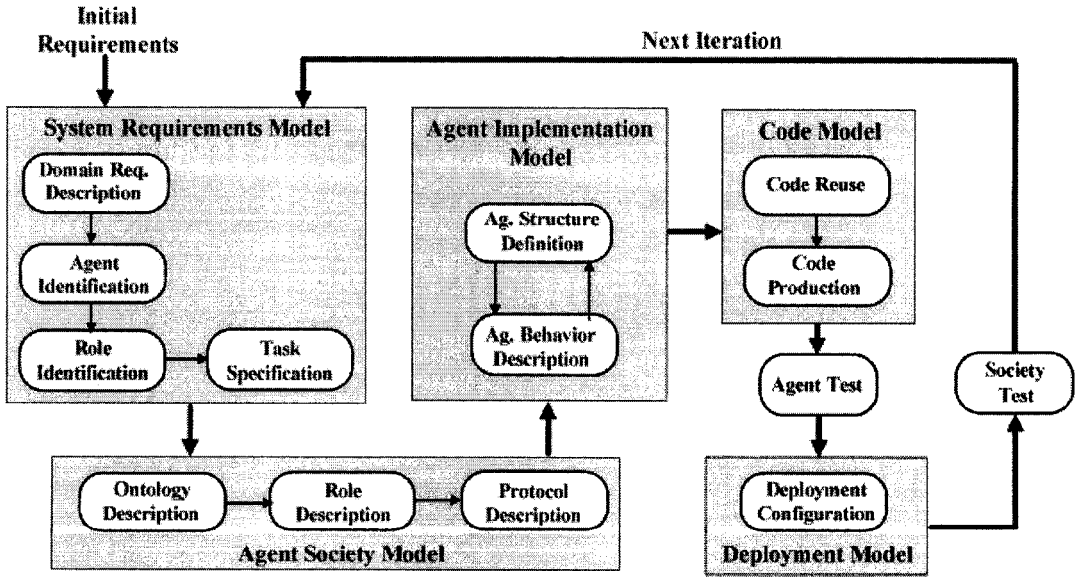


Figure 12 The models and phases of PASSI methodology [21]

As shown in Figure 12, PASSI is composed of five process components also called 'model' and each model is composed of several work activities called 'phases'.

PASSI is an iterative method such as Unified Process and other widely accepted software engineering methodologies. The iterations are two types. The first is triggered by new requirements e.g. iterations connecting models. The second takes place only within the Agent Implementation Model every time a modification occurs and is characterized by a double level iteration as shown in Figure 13. This model has two views: multi-agent and single-agent views.

Multi-agent view is concerned with the agent society (the agents' structure) in our target system in terms of cooperation, tasks involved, and flow of events depicting cooperation (behavior). Single agent view concerns with the structure of a single agent in terms of attributes, methods, inner classes and behavior. The outer level of iteration (dashed arrows) is used to represent the dependency between single-agent and multi-agent views. The inner level of iteration, which is between Agent Structure Definition and Agent Behavior Description, takes place in both views (multi-agent and single agent) and it represents the dependency between the structure and the behavior of an agent.

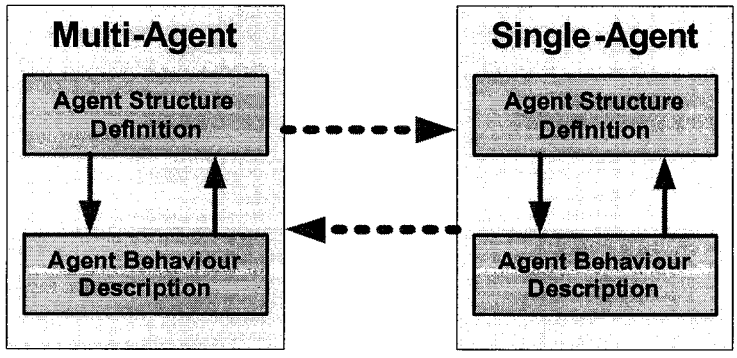


Figure 13 Agents implementation iterations [21]

As shown in Figure 12, there is also a testing activity that is divided into two phases: (single) agent test and social (multi-agent) test. In single agent test, the behavior of the agents is verified based on the original requirements of the system related to the specific agent. During the social test, the interaction between agents is verified against their cooperation in solving problems.

The models and phases of PASSI are described in the following subsections.

2.1.1 System Requirements Model

This model, as its name suggests, describes the system requirements in terms of agency and purpose and it is composed of four phases as follow:

- Domain Description
- Agent Identification
- Role Identification
- Task Specification

The Domain Description is a conventional UML use-case diagram that provides a functional description of the system. The Agent Identification phase is represented by stereotyped UML packages. The assignment of responsibilities to agents is done during this step by grouping the functionalities, described previously in the use-case diagram, and associating them with an agent. During the role identification, the responsibilities of the precedent step are explored further through role-specific scenarios using a series of sequence diagrams and the Task Specification phase spells out the capabilities of each agent using activity diagrams.

2.1.2 Agent Society Model

This model describes the social interactions and dependency among agents that are identified in the System Requirements Model and is composed of three phases as follow

- Ontology Description
- Role Description
- Protocol Description

The Ontology Description is composed of Domain Ontology Description and Communication Ontology Description. The domain ontology tries to describe the relevant entities and their relationships and rules within that domain using class diagrams. Therefore, all our agents will talk the same language by means of using the same domain ontology. In the Communication Ontology Description, the social interactions of agents are described using class diagrams. Each agent can play more than one role. The Role Description step involves of showing the roles played by the agents, the tasks involved communication capabilities and inter-agent dependencies using class diagrams. Plus the Protocol Description that uses sequence diagrams to specify the set of rules of each communication protocol based on speech-act performatives.

2.1.3 Agent Implementation Model

This model describes the agent architecture in terms of classes and methods. Unlike object-oriented approach, there are two levels of abstraction: multi-agent level and single-agent level. This model is composed of two phases as follow

- Agent Structure Definition
- Agent Behavior Description

The structure of the agent-based system is described using conventional class diagrams and the behavior of the agents (multi-agent level and single-agent level) is described using activity diagrams and state diagrams.

2.1.4 Code Model

This model is at code level and it requires the generation of code from the model using the PASSI add-in and the completing of the code manually.

2.1.5 Deployment Model

This model describes the dissemination of the parts of the agent systems across hardware processing units and their migration between processing units and it involves the following phase:

- Deployment Configuration

The Deployment Configuration describes also any constraints on migration and mobility in addition to the allocation of agents to the available processing units.

CHAPTER 3

KNOWLEDGE REPRESENTATION

Before elucidating the knowledge representation theory, we should try to clarify what exactly knowledge is in our context of data mining. According to *Webster's Dictionary* (Merriam-Webster Online) knowledge is "the fact or condition of knowing something with familiarity gained through experience or association". This definition has two parts. First, we are talking about "fact" and "condition". A fact will be such as "the sky is blue" or "the sun is hot". The second part for defining the knowledge is "experience" and "association". People gain knowledge through experience; a person put his hand in fire and he gets his hand burned. He just acquired a new knowledge by experience: fire burns. He can associate burning with fire. Since we described what knowledge is, how are we going to represent it? A natural way of doing it for a person for example will be to use a natural language like English. It is certainly not the best way to represent knowledge for computers, because natural language is inherently ambiguous. For example, two persons reading the same statement can disagree on its meaning. Therefore, we need to use some formal languages and logic to represent knowledge such as an artificial intelligence application.

As mentioned earlier, the main goal of the data mining process is to discover a priori unknown correlation. Therefore, knowledge is the output of the data mining process, the produced models. In our context of data mining, considering the intricateness of the ODM's operational data, the extracted knowledge will be much more complex than the examples above. As a result, we need a very sophisticated language to represent knowledge.

There are several languages to represent knowledge in the context of data mining. The following are one of the must accepted knowledge representation languages: KIF [23]

(Knowledge Interchange Format), PMML [24] (Predictive Model Markup Language) and RDF [25] (Resource Description Framework). In our research project, RDF will be used to represent knowledge because it is the one that is supported by the JADE platform, which will be used to implement our system.

3.1 RDF - Resource Description Framework

RDF [25] framework is based on an entity-relationship model and it is used to model meta-data. It is the selected knowledge language used in PASSI to describe the domain ontology. It is built on the rules below:

- A **Resource** is anything that can have a URI;
- A **Property** is a Resource that has a name and can be used as a property;
- A **Statement** consists of the combination of a Resource, a Property, and a value. These parts are known as the 'subject', 'predicate' and 'object' of a Statement;
- These abstract properties can be expressed in XML.

Plus, RDF is designed to have the following characteristics:

- **Independence:** Since a Property is a resource, any independent organization (or person) can invent them.
- **Interchange:** Since RDF Statements can be converted into XML, they are easily interchangeable.
- **Scalability:** RDF statements are simple, composed of three elements (Resource, Property, value), as a result they can be handled easily even the number of statements getting larger in time.

CHAPTER 4

DATABASE ISSUES

For database issues, we will focus on accessing the databases. Instead of designing our system for a specific database, Data Access Object (DAO) [26] structure should be used as shown in Figure 14.

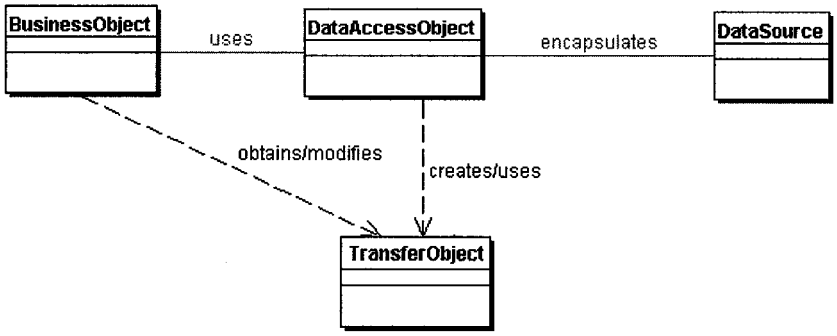


Figure 14 Data Access Object [26]

Therefore, all access to the data source will be abstracted and encapsulated. Therefore, the access mechanism required to work with the data source will be implemented by the DAO which will completely hide data source implementation details, enables transparency and easier migration, and will reduce code complexity in our system. Further details can be found in [26].

CHAPTER 5

DATA MINING PROCESS ANALYSIS

To automate the data mining process, all the steps (as shown in Figure 5) are examined and a set of solutions is proposed for each step. The detailed outcome of the data mining process automating can be found in section 4.1 of the document "*Data Crawler System Requirements and Specifications*" in APPENDIX 1. In this section, the result of this analysis will be summarized.

5.1 Step 1 – Define the problem

There is no a priori hypothesis on KD, as a result descriptive data mining is the selected approach. The set of data mining methods should be predetermined following a set of rules described in CHAPTER 6.

ODM has high dimensionality (it has more than 300 hundred variables "features") and it is a huge data repository (more than 1000 instance each day). As seen in document "*Use of clustering algorithms for knowledge extraction from high dimensional dataset*", we are faced with the curse of dimensionality caused by the high dimensionality of ODM's data. Curse of dimensionality is the fact that the demand for a large number of data samples grows exponentially with the dimensionality of the feature space. Therefore, the selected data mining methods should not be affected by this predicament.

5.2 Step 2 – Understand the data

Data understanding is about data extraction, data quality measurement and detection of interesting data subsets. This task is accomplished using a four step process [4]:

- Identifying inappropriate and suspicious attributes

- Selecting the most appropriate attribute representation
- Creating derived attributes
- Choosing an optimal subset of attributes

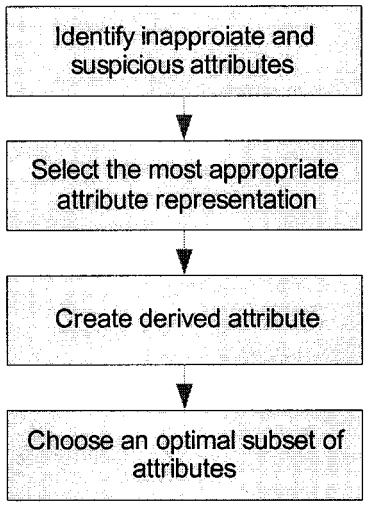


Figure 15 Data understanding steps

5.2.1 Identify inappropriate and suspicious attributes

During the data understanding step all inappropriate attributes are removed. Inappropriate attributes are described in Table II [4]:

Table II

Inappropriate attributes

Constant	Only contains a single value
Null	Has all missing values
Near Null	Has a fraction of missing values larger than a specified threshold
Many Values	Has a fraction or number of different values larger than a specified threshold.

More experimentation are necessary to establish a high-quality threshold for "near null" and "many values". At the moment in our research project, for "near null" and "many values" the specified threshold in 100% which means that all "null" and key attributes are rejected.

The suspicious attributes are described in Table III [4]:

Table III

Suspicious attributes

Type	Description
Artifact	Association and correlation with the target is greater than a specified threshold. These attributes are often unintentionally included and, without proper identification, they lead to artificially good models that do not generalize well.
Poor Predictor	Association and correlation with the target is less than a specified threshold. These attributes do not contribute much by themselves in predicting the target but combinations of these attributes may have increased predictive power. Attribute selection is the ultimate way to decide whether they are appropriate for further modeling
Near Constant	One values covers more than a specified fraction of all values
Few Values	Have less than a specified number of distinct values.
Few Cases	Have less than a specified number of distinct non-null cases.

The target attributes in the description of the first two types of suspicious attributes are the selected attributes and the sources attributes are the original attributes sets. Removal of some attributes can cause serious loss of details contained in the original attributes, therefore we need to find a balance between this loss of information and efficiency of the data mining methods. To select the best set of attributes, the loss of details can be evaluated using association measures such as mutual information, chi-squared Cramer's V and Goodman-Kruskal index proposed in [4].

For the rest of suspicious attributes, further analysis with client (Bell Canada) domain knowledge is necessary to take an action on identified suspicious attributes.

DM system should only identify the suspicious attributes without taking any further action because removing those attributes will cause loss of information and the association measures listed above can only minimize the loss of information and not eliminate it.

5.2.2 Select the most appropriate attribute representation

After identifying and rejecting inappropriate attributes, the retained attributes are processed to determine the most suitable representation. Outliers, missing values and encoding are handled during this step. For example, the continuous attributes could discretize (encoded by thresholding the original values into a small number of value range). With categorical attributes, numerous categories could be merged together. The association measures mentioned, in the precedent section, could be used to determine the optimal encoding.

This step is not realized by DCS, as more domain knowledge is needed to establish possible transformations for each attribute.

5.2.3 Create derived attributes

Attribute derivation is needed to increase the source attribute correlation with the target attribute. It is accomplished by using univariant transformation such as exponent, logarithm, quadratic function, inverse function, power function and square root. These transformations are typically only beneficial to linear regression models. Consequently, this step could be performed only to continuous attributes. The following algorithm is proposed to derivate attributes even if this step will not be implemented in this initial version.

```
for all transformations (quadratic, inverse,power,square,exp,log)
until current transformation is accepted
      Compute correlation between source and target attributes;
      Apply transformation;
      Compute new_correlation between source and target attributes;
      if new_correlation > correlation
            Transformation accepted;
            Derived attribute kept;
      else
            Transformation rejected;
```

Algorithm 1 Transformation selection to create derived attributes

This step cannot be realized by DCS because we don't have any viable reference data set with source and target attributes.

5.2.4 Choose an optimal subset of attributes

Optimal subset of attributes selection without significantly affecting the overall quality of resultant model is for reducing computational time and memory requirements. As mentioned in our objectives, computational time and memory are not an issue in this project. Therefore, no more transformations will be applied on previously selected attributes. This will also simplify the design.

There are two algorithms suggested in [4] for attribute selection: *expectation of Kullback Leibler distance* (KL-distance) and *Inconsistency Rate* (IR). Those two algorithms can be used in future implementations.

5.3 Step 3 – Prepare the data

This step presents the activities needed to construct the final dataset for modeling. In our case, this step prepares the dataset for DM methods used for modeling. The data should be previously cleansed by the "understand the data" step.

For example, clustering algorithms of WEKA can only be applied to numerical or categorical data. Other type of data such as string needs to be transformed to categorical data otherwise they are excluded.

5.4 Step 4 – Estimate the model

During this phase, the DM algorithms are applied to the prepared dataset. Our goal of applying KD on data from ODM is to get a better understanding of it. As a result, the selected approach is descriptive data mining. And, the DM algorithms that produce descriptive models are used. As mentioned during the precedent sections, use of a fixed set of DM algorithms will allow us to automate this process. The following type of descriptive DM algorithms could be used.

- Decision Trees
- Association Rules
- Clustering

This list isn't exhaustive. For example, PART algorithm that we selected to study the impact of high dimensionality was indeed a neural network algorithm adapted for clustering.

5.4.1 "Estimate the model" process flow

The modeling process (and all our DM system) is designed using Common Warehouse Metamodel (CWM) specification v1.1 chapter 12 and Java Specification Request 73: Java Data Mining (JDM). The modeling involves a four-step process as shown in Figure 16:

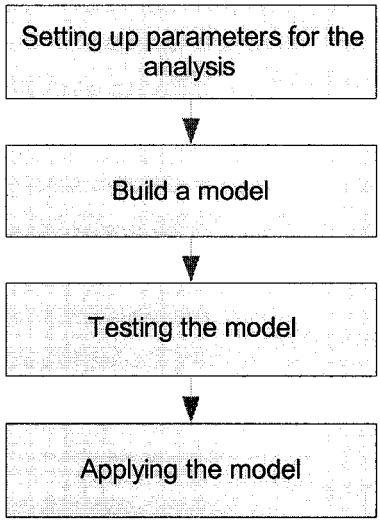


Figure 16 Estimate the model step

A brief description of each step is below:

- *Setting up parameters for the analysis*: During this step all parameters or inputs that affect model building are set. The parameters or inputs values are predetermined. A detailed analysis is necessary to identify the parameters for each model.

- *Build a model*: A model is build. The model is a compressed representation of input data and it contains the essential knowledge extracted from data.

- *Test the model*: Model testing estimates the accuracy of the model. It is processed after model building. The inputs are the model and a data sample.

- *Apply the model*: Model applying is used to make predictions. Since we are doing unsupervised data mining, apply will produce probability of assignment. For example, with clustering, apply assigns a case to a cluster, with the probability indicating how well the case fits with a given cluster.

5.4.2 Decision Trees

Decision tree is a hierarchical representation that can be used to determine the classification of an object by testing its values for certain properties [6]. Besides the fact that decision tree is a descriptive model, the tree's hierarchical structures are easy to understand and effectively explore the model.

5.4.3 Association Rules

Association rules (also called "market basket analysis") are about finding relation between objects such as X=>Y where X and Y are sets of items and "=>" is a relation. For example, "if some one buy a chocolate is likely to buy a candy bar" is an association rule. Those relations are useful for data explorations.

5.4.4 Clustering

Clustering aims to partition a given dataset into several groups with records similar to each other. There are various clustering methods. Iterative, incremental and hierarchical clustering are the most popular. In our project, incremental and hierarchical clustering will be favored against Iterative clustering for several reasons.

First, hierarchical nature of the resulting model is easy to understand. And, these methods are very flexible concerning the distance metric used to group the records, making the hierarchical clustering methods more easily adaptable to a different problem: more easy to automate the process. Another advantage of hierarchical clustering methods is their computational complexity (even if we didn't considered it as a primordial objective) requiring either $O(m^2)$ memory or $O(m^2)$ time for m data points.

When the dimensionality is very high, as in our case, incremental clustering method generates explicit knowledge structure that describes the clustering in a way that can be

visualized. For iterative algorithms this is possible only if the dimensionality is not too high.

5.5 Step5 - Interpret the model

This step is realized by the user of the DM system. The problem of interpreting the resultant models is very important since a user does not want hundreds of pages of numeric results. It will be difficult to make a successful decision from data mining results that are not presented in a way that is easy to understand to the user. For all these reasons we decided to define this step explicitly.

This step is about the presentation of the resulting models in a format that will be easily understandable by the user. The user of the system is the Bell Business Intelligence and Simulation team. In our implementation we should focus on the adaptability of the format to the user needs instead of using a static format, since user needs change over time.

CHAPTER 6

DATA MINING METHODS SELECTION

In this section, instead of listing the selected DMTs we will explain the strategy for selecting data mining methods for the Data Crawler System. The selection strategy is the outcome of the analysis that we did during our methods selection process. The study mostly focused on the impact of high dimensionality of the dataset on the quality of the produced model with several clustering methods. This analysis is described in detail in document "*Use of unsupervised clustering algorithm in high dimensional dataset*" in APPENDIX 2.

Three clustering algorithms were used to conduct the analysis: PART algorithm, fuzzy c-means and k-means. The clustering algorithms were selected because they are unsupervised DMTs and descriptive data mining can be performed only with unsupervised DMTs. Those DMTs were applied on synthetic data similar to data from Bell's ODM. The synthetic data was generated randomly. The data from Bell's ODM weren't used because we don't have any valuable prior knowledge on it and a reference is needed to validate the produced clusters.

In high dimensional data set two points will be far apart from each other on few dimensions only and on full dimensional space and on full dimensions the distance between every pair of points will be almost the same. Therefore, as it has been confirmed by our experimentation, most of the classical clustering methods will not be efficient for mining data from Bell's ODM because of the sparsity of data in high dimensional space and all dimensions are used by these DMTs. As seen with k-means and fuzzy c-means results, searching clusters in full dimensions can lead to erroneous results. Subspace clustering methods are an excellent way of mining high dimesional data.

Before deciding on a DMT several points should be considered. For example, fuzzy c-means and k-means need the number of searched clusters to be specified. In our case there isn't enough prior knowledge to identify the number of searched clusters.

Therefore, the following criteria should be considered in selecting a method for our system:

- *Prior knowledge requirements and domain specific strategy of DM methods:*

  Prior knowledge requirements are the most important aspect to consider for choosing a method for data mining because of our lack of knowledge about data. For example, if the method to be selected requires any knowledge related to clusters such as number of clusters or number of dimensions (or the specific dimensions) that form the clusters or any other information specifying the form of the clusters, the method should be rejected.

  Otherwise, the input parameters of the method should be identified and the value of the input parameters should be defined according to our data. The domain specific strategy of data mining methods is the establishment of a strategy for defining the prior knowledge requirements of data mining methods according to input data space.

- *Unsupervised, Descriptive data mining methods:*

  The goal in building the data crawler system is to gain an understanding of Bell's operations by uncovering patterns and relationships in ODM. In literature, this is know as descriptive data mining which produces new, nontrivial information based on the available data set.

  Unsupervised learning's goal is to discover "natural" structure in the input data and there is no notion of output during the learning process compared to supervised learning where unknown dependencies are

estimated from known input-output samples. For example, with classification analysis which is a supervised learning, the output is taken from a set of known class but with clustering which is an unsupervised learning, we don't have a set of a priori known clusters.

Therefore, only unsupervised descriptive data mining methodologies such as clustering analysis, association rules, or some artificial neural networks such as PART algorithm will be used.

- *Sensibility to high dimensionality:*

  As shown in [27], in a high dimensional space, the distance between every pair of points is almost the same for a broad variety of data distributions and distance functions. For example, in such condition, we can't perform clustering in full space of all dimensions. The subspace clustering which aims to find clusters formed in subspaces of the original high dimensional space is a viable solution to the problem.

  Therefore, we will opt for data mining methods that are not affected by the high dimensionality of data.

- *Scalability:*

  Scalability of data mining methods isn't a priority requirement because our system isn't online and the quality of the extracted knowledge is more important than the responsiveness of the system. However, it is an aspect to consider in design because there exist some methods which offer a good balance between performance and output quality such as MAFIA [28] clustering method. This algorithm uses an adaptive grid based on the distribution of data to improve efficiency and cluster quality and introduces parallelism to improve scalability.

- *Complexity (processing time):*

Same as with scalability, the complexity isn't a priority requirement but it will be considered in design and during the selection of the data mining methods the same way as scalability.

CHAPTER 7

DATA CRAWLER ARCHITECTURE

Most previous chapters contents are results of the literature survey; all tools and concepts that are needed to build the data crawler system are presented, except the CHAPTER 5 that resumes the analysis to automate the data mining process and the CHAPTER 6 that portrays the results of the study conducted to select the DMTs for DCS system. From here, the presented artifacts and works are outcomes of this current project.

In this section, a high level architecture of the system is presented. Detailed information on the Data Crawler System design and architecture can be found in the document "*Data Crawler System Requirements and Specifications*" in APPENDIX 1.

The data crawler architecture is based on the generic agent application architecture proposed in [11], as shown in Figure 17.

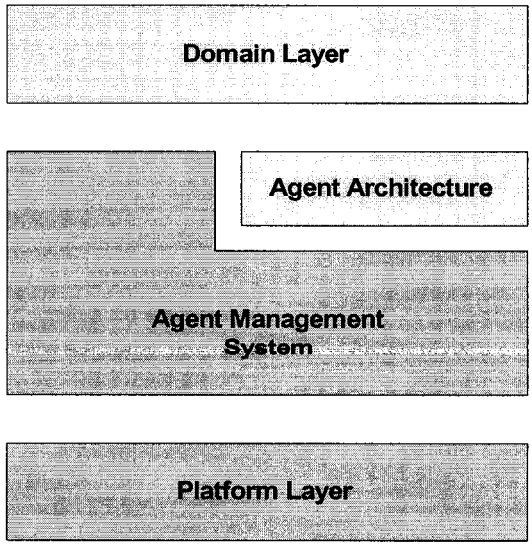


Figure 17 A Generic Agent-based System Architecture

In Figure 17, the Platform Layer corresponds to the hosts where the target system will run. The Agent Management System represents a crossing point between agents and the platform and the Agent Architecture implements the runtime environment. The Domain Layer relates the domain specific aspects.

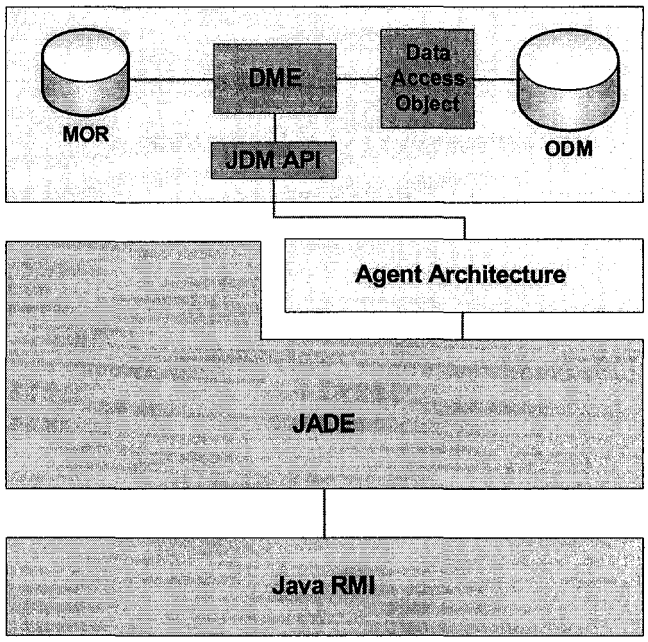


Figure 18 The Data Crawler Architecture

JADE is selected as the Agent Management System, which will run on Java RMI. Considering that data mining is our domain and the data mining aspect of our system is designed using JSR-073: JDM [14], then the Domain Layer will be designed using the data mining architecture proposed in [14].

The data mining aspect of the system can be realized by any data mining engine that complies with JDM.

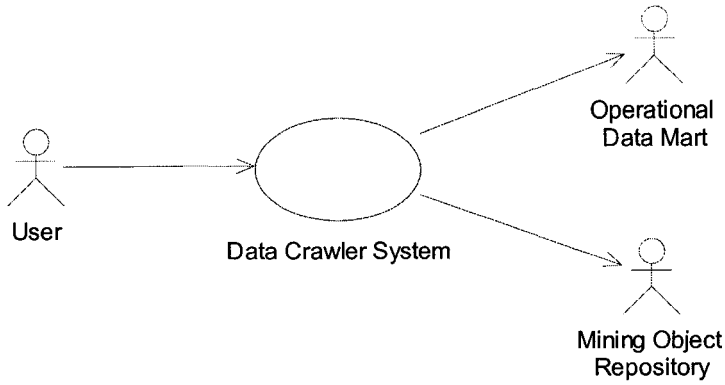


Figure 19 Context Diagram

Essentially, the Data Crawler System (DCS) has three actors: user, Operational Data Mart (ODM) and Mining Object Repository (MOR). *User* is the data mining clients; people from Bell Business Intelligence and Simulation teams or managers who are in decisional context. User should access DCS to visualize produced models and to request for specific data mining tasks (e.g. applying a model produced by the DCS on data set specified by the user). *Operation Data Mart (ODM)* is the database where all data to be mined are picked up. The DCS should load data from ODM and should apply its data mining technique on this data and should save the results (e.g. models, statistics, apply results, etc.) and other mining objects (e.g. build tasks, build settings, algorithm setting, etc.) to the *Mining Object Repository (MOR)*.

The entire detailed functional description of DCS is given in Figure 20.

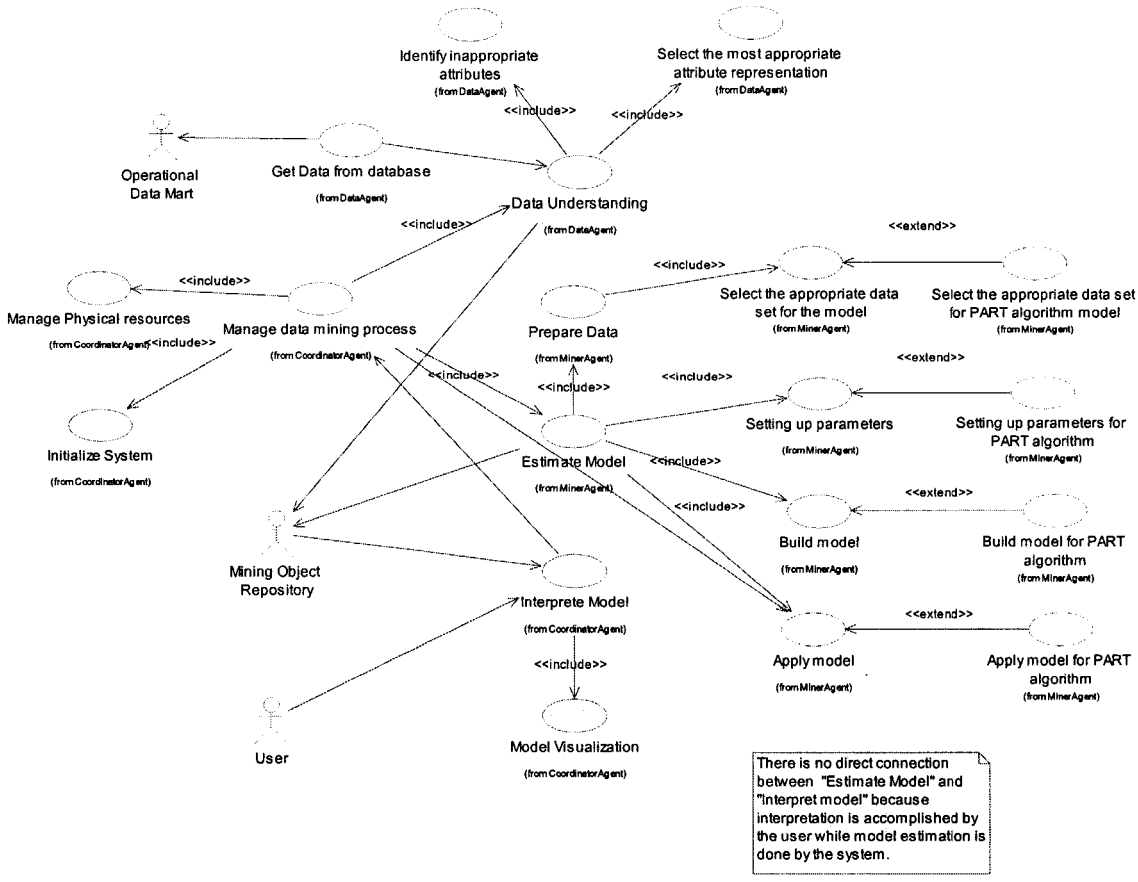


Figure 20 Domain Description

The domain description is based on the data mining process as shown in Figure 5 and the functional analysis to automate each step of the DM process is described in CHAPTER 5.

The data mining process begin after the system initialization. Then, DCS get data from ODM and realize "understand the data" step of the DM process and the results (cleansed data) are saved to MOR. Then, the "estimate model" step starts, but "prepare data" step will be started first, since it is considered as part of the "estimate model" step; this is

why there is no direct link between "manage data mining process" use case and "prepare data" use case.

Once the "prepare data" step is finished, a model is build and the resulting model is saved in MOR. Then, the user can interpret the produced model, by visualizing the model and/or applying the model on other data.

DCS system should realize first three steps ("Understand data", "Prepare data" and "Estimate model") of the DM process in loop as long as there is available data in ODM and there is available resource (processing unit) as well. The last step "interpret model" is realized by DCS on user requests.

Basically, three agents have been identified as shown in Figure 21:

- Data agent,
- Miner agent,
- Coordinator agent.

The diagram in Figure 21 shows the structure of each agent and their relation to each other and the actors. Each agent is represented by a class and its tasks are shown as operations.

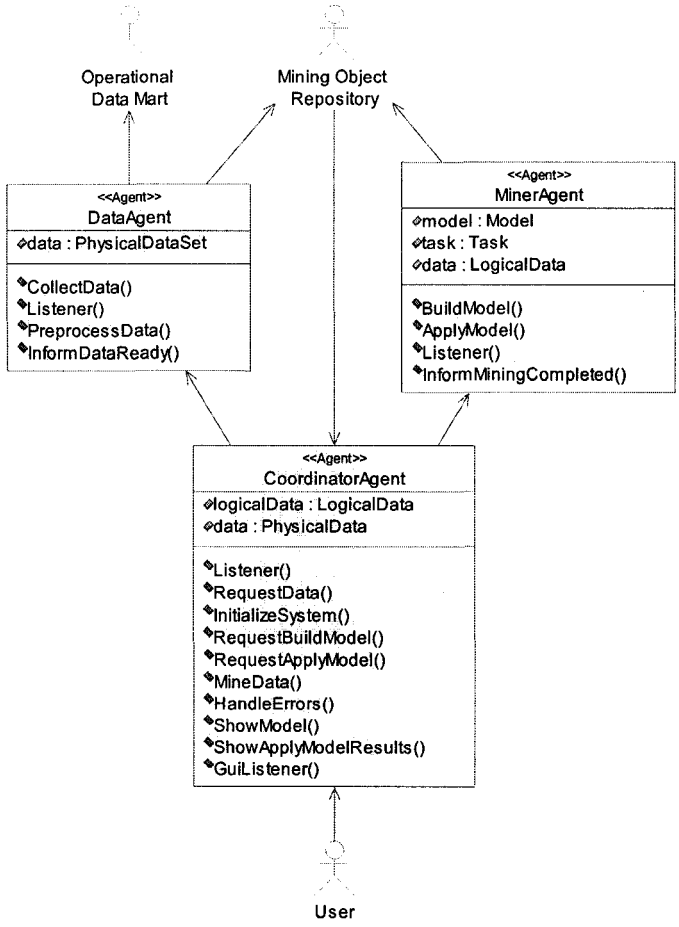


Figure 21 Agents Structure Definition Diagram

Data agent acquires data from ODM database and accomplishes the "data understanding" step which consist of data extraction and data quality measurement and detection of interesting data subsets.

Miner agent will prepare the previously cleansed data for the data mining algorithm and will produce a model that will be saved on MOR. This agent will also apply models on data following user's instructions.

The Coordinator agent has mainly four tasks:

1. *Physical resources management:* In our context, the physical resources are the host on which the agents run. Managing activity consists of deciding on which node which agent will be executing. The resources management should be as follow:

For each node, there should be only one Data agent and one Miner agent executing.

   There should be only one coordinator agent on the system. In our implementation the physical resources management is implicit, which means that the coordinator agent don't take any particular actions to realize it. At the start-up, each agent is created according to the rule above. The coordinator agent should activate or deactivate the data and miner agents depending on the available jobs. At this level the Physical resources management is too simple but eventually with more complicated mining strategy or with limited resources, this separated functionality will be important.

2. *Data mining management:* The data mining management consists sequentially of
   - Requesting data agent to get and prepare data for mining,
   - Requesting miner agent to prepare and mine the cleansed data and
   - Keeping track of the data mining process within ODM's data. All data will be mined in sequence.

3. *Error management:* At this level it consists of logging the errors that occur.

4. *User interaction:* It will allow users to visualize resulting models as described in section 4.1.5 and apply models on user defined data set. Eventually, other type

of task could be integrated to the system that will facilitate the model interpretation step.

CHAPTER 8

EXPERIMENTS AND RESULTS

The experimentation was mostly concentrated on the impact of the high dimensionality of the data on the quality of the extracted knowledge. Detailed information on the conducted analysis can be found in document "*Use of unsupervised clustering algorithm in high dimensional dataset*" in APPENDIX 2. In this section the results are summurized.

Three clustering algorithms were used: PART, Fuzzy c-means and k-means. PART [30] is a subspace clustering method effective with high dimensional data sets. Fuzzy c-means (FCM) is a procedure of clustering data wherein each data point belongs to a cluster to a certain degree that is specified by a membership grade. This technique was originally introduced by Jim Bezdek in 1981 [29]. The algorithm of k-means [31] is a classical clustering method.

The PART algorithm requires two external parameters: vigilance parameter (ρ) and distance vigilance parameter (σ). The distance vigilance parameter is related to the range of the value of each attribute (distance between two points) and it was set to 10 because same range of value for each point in synthetic data was used as in [30]. The vigilance parameter indicates the number of dimension on which the distance between each point is evaluated. For vigilance parameter, we should choose a value low enough to find clusters of any size and high enough to eliminate the inherent randomness in data set with high dimensionality. Usually, there isn't correlation in large number of random data points with large set of dimensions. After few preliminary experimentation to find the best value for vigilance parameter, high dimensional subspace clusters (~270-dimensional clusters) were best discovered with ρ=13 and low dimensional subspace

clusters (~10-dimensional clusters) were best discovered with ρ=3. In case of data set with clusters varying within full range subspace ρ=6 gave the best results.

Thus, PART algorithm was successful in finding the exact number of clusters with their exact entries when the clusters had high dimensional-subspace. Even the outliers were found. However, when the input clusters had low dimensional-subspace, the accuracy of the output clusters was majorly affected, because the value of the vigilance parameter (ρ=3) wasn't high enough to eliminate the randomness within the data set. The highest accuracy that the output clusters had was 78%, most of the other clusters accuracy was around 60%. When the input data set had clusters with dimensional size varying within full range dimensional space, most clusters (9 of 10) were found with very high accuracy (~100%). One cluster couldn't be extracted because its center was very close to another cluster and it had very low number of sample compared to other clusters.

With FCM the experimentation was conducted differently than PART algorithm because it is known that PART is proficient with high dimensional data with subspace clusters but it is not the case for FCM if it will be able to extract subspace clusters from high dimensional data or not. Instead of applying FCM directly on high-dimensional data set, the number of dimensions of the input data set and observed if the FCM was still able to detect input clusters.

FCM could only find clusters defined in full-dimensional space. Full-dimensional space means that if we have a data set with 4 dimensions (i.e. attributes), the clusters must be defined using all the 4 dimensions otherwise FCM can't detect them; FCM isn't suitable for extracting subspace clusters. Also, when input data set had 10 dimensions or higher FCM wasn't able to find the input clusters.

This experimentation also confirmed that k-means isn't suitable for extracting subspace clusters. Only 2 data sets of 5 dimensions were used with k-means algorithm. In the first

data set the clusters were defined using all dimensions and in the second data set the dimensional size of the clusters varyied between 2 and 5. In case of data set formed with different dimensional size clusters none of the input clusters were identified while with data set containing full dimensional clusters most clusters were found but none existing cluster coming from outlier data also was found and some clusters were identified as one cluster.

One of the most valuable outputs of our experimentation was the establishment of a strategy for selecting data mining methods for DCS as described in CHAPTER 6.

We were not successful in testing our agents doing data mining since we couldn't find any DME in accordance with JDM's API.

CONCLUSION AND FUTURE WORK

The most prominent difficulty that we faced during this project was the fact that data preparation for applying DMTs was very difficult to realize with real data. The data set given by Bell Canada contains several errors and typos even if it was cleansed before. Most attributes had missing values, in some case missing values represent more 70% of the total instances and some DMT can't be used if input data set has missing values. The data preparation step is as important as applying DMTs even more since the pruned data set can affect the created model accuracy and validity. Also, the data preparation step is a very time consuming step (~80% of the time allocated to DM process), it is more valuable to automate the data understanding and data preparation phases than the model estimation phase.

The selection of the DM methods was also another source of frustration since we couldn't build useful models by any DM methods because of several reasons. First, the descriptive data mining approach, limited our selection of DM methods to unsupervised DM methods. Another reason was that we were confronted with the curse of dimensionality caused by the high dimension of the original dataset. A classical solution to this problem is the reduction of the dimension of the original dataset. To do that, we need to have some a priori information about the data, which we didn't have. Therefore, we need to select our DM methods from a set of methods that can deal with the curse of dimensionality, such as PART algorithm. A DM methods selection strategy for DCS is established as described in CHAPTER 6.

More analysis need to be done to optimize the quality of the pruned data, which requires more implication of the client (Bell Canada). The points where the client can intervene are detailed during the functional analysis of the data understanding step. Also we could select several DM methods in accordance with the selection strategy established in document "*Use of clustering algorithms for knowledge extraction from high dimensional*

dataset" and test them. The must prominent data mining techniques will be hierarchical clustering, temporal data mining methods. Most features (data attributes) in ODM's data are date type (temporal type) and categorical type. Therefore, future studies with data mining methods could be done with the temporal algorithms.

After analyzing the problem domain (data mining), we began designing the agent-based system. We needed to select proper agent-based system designing and development methodology and tools that will diminish (make it transparent to the developer), the inherent complexity of the agent technology. Since the field of agent-based and multiagent systems is still in its infancy, we couldn't find appropriate tools and most of the tools or methodologies are incomplete or not mature enough to be used. We began our design with the MASSIVE [11] methodology, an academic project, but after using it, this methodology revealed to be very difficult to apply in a real world project and its documentation wasn't easily understandable. Therefore we changed for the PASSI methodology which was more straightforward and easily applicable thanks to the PTK toolkit, which is an extension to Rational Rose software that implements the PASSI methodology. We faced the same kind of difficulties during the selection of the agent platform. After trying several agent development toolkits (ZEUS, FIPA-OS, JADE), we ended with the JADE development platform.

In our design, all data mining logics are encapsulated into a DM engine in conformity with the JDM specifications that agents accessed to realize their data mining tasks. This greatly improves the modifiability of the system. Therefore, we could easily update/modify/add the data mining algorithm or tasks without or use any data mining engine from different vendors as long as they respect JDM API without affecting the agents' implementation. However, the only DM engine in accordance with JDM API that has been found was the JDM implementation provided with JDM specifications. This engine was made of stubs (empty methods that don't contain any code or temporary code to simulate some behavior) that we discovered later. Therefore, we couldn't use it

in our project. There was also a JDM implementation provided by KXEN Inc. (as yet the only one that exists that we know), but it wasn't publicly available. As a result, we couldn't experiment the interaction of our agents with the DM engine. Considering the complexity of a data mining engine, developing a DME of our own could be a great project that will complement DCS framework. The lack of an available DME agent made some questions not answered: Considering that the agents will be spread out across several hosts and they connect to the DME to execute their DM tasks, the DME can be implemented as a server and agents and the DME can have a client-server relationship. But we could also implement it as a small component that the agents carry out with them containing only the services that they needs (e.g. data agent will have a DME that provides only services related to data preparation and the miner agent will have a DME that provides only services related to model estimation related tasks). More experimentation related to DME is needed to determine the best solution to our problem.

JDM leaves to the developer's discretion how the DME and MOR are implemented as long as they are in accordance with JDM API. The same type of questions as with DME rises up when it come to MOR implementation. MOR can be implemented as a huge database that contains all the mining objects and each agent accesses it through DME. We can also implement it in a layered fashion.

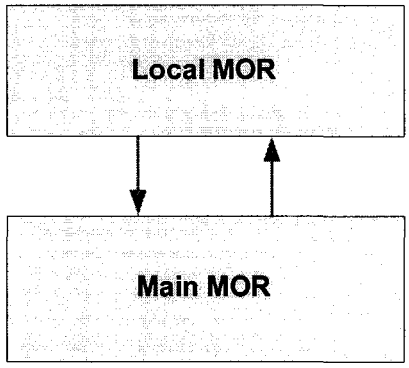


Figure 22 MOR Layered Structure

One layer could be local to the agent and every time an agent produces a mining object it will be saved to the local layer of MOR. Then, the mining objects in local layer are systematically saved to the main MOR. Main layer serves as a backup that every agent can access and when an agent needs a mining object located in the main MOR then it will be loaded to the local MOR. This approach could be more complex but also more efficient. Therefore, more experimentation is necessary to study the effect of the distributed aspect of the systems on the agents and MOR relations, MOR's internal structure and other predicament that could be engendered.

Mobile agents can be used to verify their impacts on data mining process and to have a deeper understanding of the potential of using agents in data mining. Another facet that could be further analyzed is the distributed data mining where the computation can be spread out through several nodes.

The data mining process is an iterative process where every time the results of the current step are not satisfactory we go back to the step before and iterate as long as the results are acceptable. In our implementation, each step is considered being done once; therefore the iterative aspect of the data mining process was neglected. To evaluate the results of a step metrics are needed and they can't be obtained without a priori information. For example, to test the quality of a produced model, it should be compared to a reference model. Considering that we were doing descriptive data mining without any a priori information, it wasn't possible to have any measures. Therefore, more study with the implication of the client is necessary to establish some measures to evaluate produced mining objects. Once we have the ability to evaluate the produced model, the system could be improved to be more intelligent and dynamic. For example, when the miner agent produce a model and inform the coordinator agent of its results, the coordinator agent could refuse the produced model and reschedule the build model task with modified input build setting parameters.

Another aspect of the system that could be improved over time is the GUI. The user could have more control over the data mining process.

APPENDIX 1

Data crawler system requirements and specifications

1. INTRODUCTION

This document describes the software requirements and design details of the Data Crawler System (DCS). The DCS is designed and developed using PASSI (Process for Agent Societies Specification and Implementation) methodology, which is a step-by-step requirement-to-code method.

1.1 Purpose

The purpose of DCS is to accomplish data mining autonomously, which will reduce the complexity of applying data mining process, consequently will reduce inherent cost related to the complexity. Considering that the success of a data mining system is caused mainly by its adaptability to the client's domain, our system should be easily extensible and new data mining algorithms should be inserted into system with a small amount of effort. To adapt the data mining algorithms to the client's domain, the system should provide tuning possibilities to the user.

The system should provide to the user the ability to see produced model by DCS, apply those model on data specified by the user and compute some statistics on attributes.
This document also describes nonfunctional requirements, design constraints and other aspects of the system necessary to provide a complete description of the Data Crawler System.

1.2 Scope

This document covers all features of DCS system. Basically the system features are as follow: the system should access Operational Data Mart (ODM) database and get a data subset and should apply data mining algorithms on the data subset and save the resulting

models into Mining Object Repository (MOR). The system should show produced model to the users and allow users to apply those models on user specified data.

1.3 Overview

Considering that the DCS system is designed and implemented using PASSI methodology, the structure of this document will be mostly influenced by the structure of the PASSI methodology. Before getting into PASSI models, an overall description of the system is given followed by the architecture of the DCS. Then, the system requirement model, Agent Society model, Agent Implementation model, Code Model and Deployment model are presented. Finally, the interfaces of DCS are described.

2. OVERALL DESCRIPTIONS

In this section, design constraints and non-functional attributes not present in the design constraints section are presented.

2.1 Design Constraints

In this section, all design constraints such as used software language, components, development toolkits and class libraries are described.

2.1.1 Software Development Language and Libraries

The system is developed with Java programming language using Java Specification Requirement (JSR) 73: Java Data Mining (JDM) v1.0, Java API for data mining capabilities of the system and using Java Agent Development Framework (JADE) for implementing the agent technology.

2.1.2 Agent Development Methods

Since agents are still a forefront issue, there are many methodologies proposed and all of them have interesting features and capabilities. Most of them are still in beta version and don't have a widespread acceptance in the agent community. The following diagram shows all existing agent-oriented methodologies and their relations to each other.

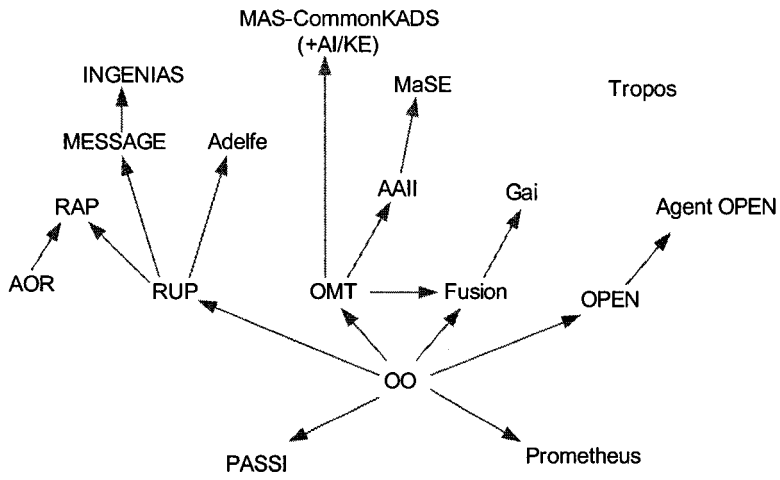


Figure 23 Genealogy of Agent-Oriented Methodologies [21]

Our goal in this project isn't to test and review all existing agent-oriented (AO) methodologies but to use one of them. In first place, we selected MASSIVE [11] (Multi-Agent SystemS Iterative View Engineering) and begin developing our system using this methodology (we couldn't find any other methodology at the moment and we thought this method was alone in its category). This method is based on a combination of standard software engineering techniques and it features a product model to describe the target system, a process model to construct the product model and an institutional framework that supports learning and reuse over project boundaries.

This methodology is an academic project and it doesn't have a widespread acceptance and use (as we can observe it isn't listed in the Figure 23). After using it, MASSIVE methodology revealed to be very difficult to apply in a real world project and its documentation wasn't easily understandable. Therefore, we continued our research for finding another agent-oriented methodology.

Therefore we found PASSI methodology, which was more easily applicable and implemented in a toolkit PTK, which allows the design of an agent-based system using

Rational Rose with the methodology. PASSI is described in the section 2.1.2.1 and PTK is described in section 2.1.3.1.

2.1.2.1 PASSI

PASSI [40] (Process for Agent Societies Specification and Implementation) is a methodology for designing and developing multi-agent systems, using a step-by-step requirement-to-code process. It integrates the design models and concepts, using the UML notation, from two dominant approaches in the agent community: OO software engineering and artificial intelligence.

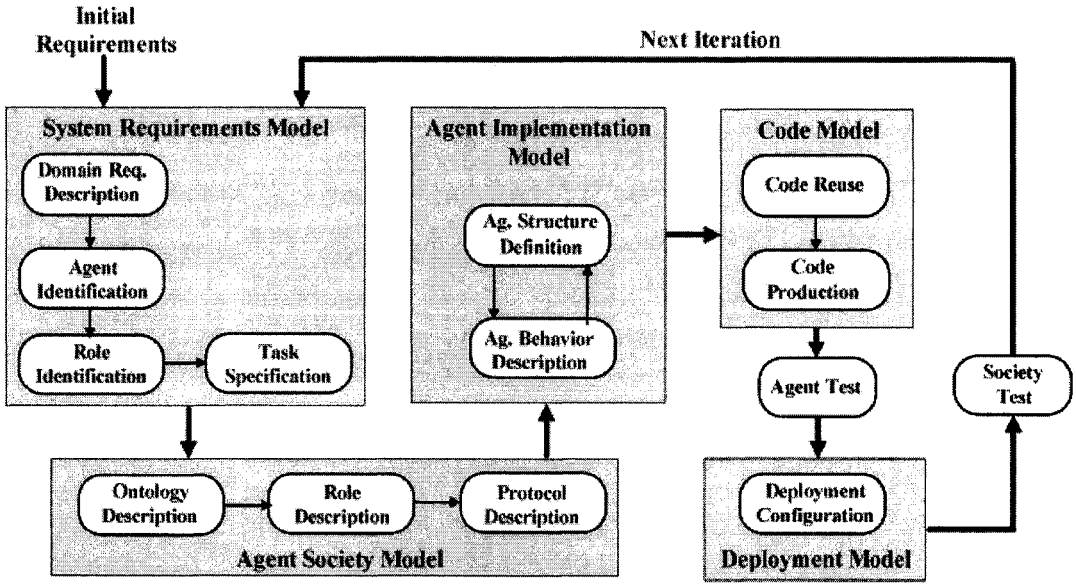


Figure 24 The models and phases of PASSI methodology [21]

As shown in Figure 24, PASSI is composed of five process components also called 'model' and each model is composed of several work activities called 'phases'.

PASSI is an iterative method such as Unified Process and other widely accepted software engineering methodologies. The iterations are two types. The first is triggered by new requirements e.g. iterations connecting models.

The second takes place only within the Agent Implementation Model every time a modification occurs and is characterized by a double level iteration as shown in Figure 25. This model has two views: multi-agent and single-agent views. Multi-agent view is concerned with the agent society (the agents' structure) in our target system in terms of cooperation, tasks involved, and flow of events depicting cooperation (behavior). Single agent view concerns with the structure of a single agent in term of attributes, methods, inner classes and behavior. The outer level of iteration (dashed arrows) is used to represent the dependency between single-agent and multi-agent views. The inner level of iteration, which is between Agent Structure Definition and Agent Behavior Description, takes place in both views (multi-agent and single agent) and it represents the dependency between the structure and the behavior of an agent.

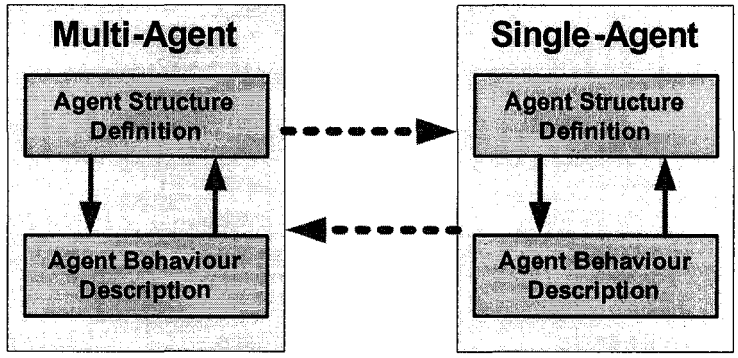


Figure 25 Agents implementation iterations [21]

As shown in Figure 24, there is also a testing activity that is divided into two phases: (single) agent test and social (multi-agent) test. In single agent test, the behavior of the agents is verified based on the original requirements of the system related to the specific

agent. During the social test, the interaction between agents is verified against their cooperation in solving problems.

The models and phases of PASSI are described in the following subsections

2.1.2.1.1 System Requirements Model

This model, as its name suggests, describes the system requirements in terms of agency and purpose and it is composed of four phases as follow:

- Domain Description
- Agent Identification
- Role Identification
- Task Specification

The Domain Description is a conventional UML use-case diagram that provides a functional description of the system. The Agent Identification phase is represented by stereotyped UML packages. The assignment of responsibilities to agents is done during this step by grouping the functionalities, described previously in the use-case diagram, and associating them with an agent. During the role identification, the responsibilities of the precedent step are explored further through role-specific scenarios using a series of sequence diagrams and the Task Specification phase spells out the capabilities of each agent using activity diagrams.

2.1.2.1.2 Agent Society Model

This model describes the social interactions and dependency among agents that are identified in the System Requirements Model and is composed of three phases as follow:

- Ontology Description
- Role Description

- Protocol Description

The Ontology Description is composed of Domain Ontology Description and Communication Ontology Description. The domain ontology tries to describe the relevant entities and their relationships and rules within that domain using class diagrams. Therefore, all our agents should talk the same language by means of using the same domain ontology. In the Communication Ontology Description, the social interactions of agents are described using class diagrams. Each agent can play more than one role. The Role Description step involves of showing the roles played by the agents, the tasks involved communication capabilities and inter-agent dependencies using class diagrams. Plus the Protocol Description that uses sequence diagrams to specify the set of rules of each communication protocol based on speech-act performatives.

2.1.2.1.3 Agent Implementation Model

This model describes the agent architecture in terms of classes and methods. Unlike object-oriented approach, there are two levels of abstraction: multi-agent level and single-agent level. This model is composed of two phases as follows:

- Agent Structure Definition
- Agent Behavior Description

The structure of the agent-based system is described using conventional class diagrams and the behavior of the agents (multi-agent level and single-agent level) is described using activity diagrams and state diagrams.

2.1.2.1.4 Code Model

This model is at code level and it requires the generation of code from the model using the PASSI add-in and the completing of the code manually.

2.1.2.1.5 Deployment Model

This model describes the dissemination of the parts of the agent systems across hardware processing units and their migration between processing units and it involves the following phase:

- Deployment Configuration

The Deployment Configuration describes also any constraints on migration and mobility in addition to the allocation of agents to the available processing units.

2.1.3 Development toolkits

The data crawler system should be developed using the following toolkits

- PTK 1.2.0 (PASSI Tool Kit)
- JADE 3.3 (Java Agent DEvelopment Framework)

2.1.3.1 PASSI Tool Kit

PASSI Tool Kit (PTK) is a compilation of two tools that interacts with each other. The first one is *PASSI Add-in*, which is an extension to Rational Rose software that implements the PASSI methodology. Thus, the user can follow the PASSI's phases easily using automatic diagrams generation feature of the tool.

The second tool is *Pattern Repository*. It is an interface for managing a repository of patterns. It supplies the repository of patterns and the user can pick up and integrate patterns in their MAS being developed, using a search engine.

2.1.3.2 Agent Platform

Before getting in to Java Agent Development Framework (JADE), we will briefly describe available agent development platforms and the taken approach for selecting it. There exists panoply of agent platforms and toolkits [41]. They have different level of maturity and quality. Therefore, we established a set of criteria for selecting the one that is most suitable to our project. The agent platform was selected according to following criteria:

- *Standard compatibilities:* The agent platform must be in conformity with FIPA standards. FIPA (the Foundation for Intelligent Physical Agents) is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies.
- *Communication:* The agent platform must support inter-platform messaging
- Usability and documentations: The documentation must be clear, easy to understand and free of bugs. Also, there should be enough examples and tutorials to run and test the platform.
- *Availability:* The agent platform must be publicly available.
- Development issues:
    - The agent platform must be supported by PTK, used for designing and developing our system.
    - The agent platform must be coded in Java.
    - The agent platform has an active development community
    - The agent platform has a widespread acceptance in the agent communities.

Our choice was narrowed on FIPA-OS and JADE, considering that PTK generates java code from UML diagrams for those platforms only. We also evaluated ZEUS Agent Building Toolkit which is an integrated development environment for creating multi-agent systems. All those three agent platforms support FIPA standards and inter-platform messaging and they are publicly available.

Like FIPA-OS and JADE, ZEUS provides support for development of FIPA compliant agents. ZEUS provides a runtime environment, which facilitate applications to be monitored and other tools like reports tool, statistics tool, control tool, society viewer and agent viewer that make it an excellent agent development platform. However, the documentation is very weak that make it difficult to run applications on it. Plus, it is not supported by PASSI Add-in. Therefore, we will not provide any further details on this platform.

2.1.3.2.1 FIPA-OS

FIPA-OS is a component-based toolkit enabling rapid development of FIPA compliant agents. In Figure 26, the core components of the FIPA-OS are illustrated. The FIPA Agents exist and operate within this normative framework provided by FIPA [36]. Combined with the Agent Life cycle, it establishes the logical and temporal contexts for the creation, operation and retirement of Agents.

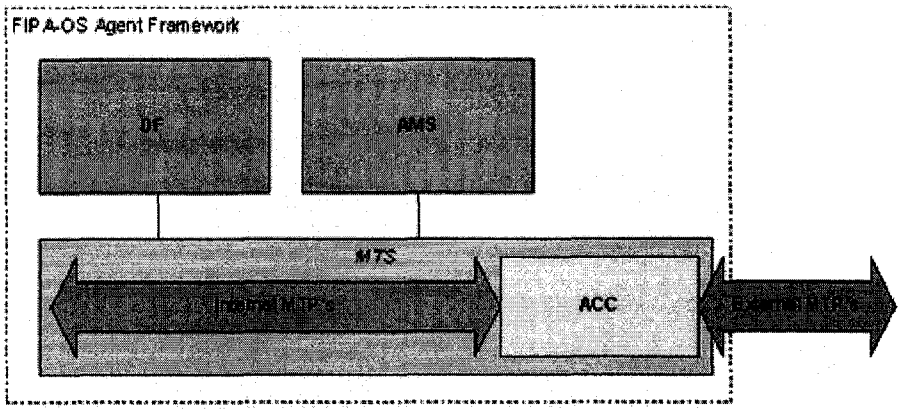


Figure 26 FIPA Reference Model [34]

The Directory Facilitator (DF) and Agent Management System (AMS) are specific types of agents, which support agent management. The DF provides "yellow pages" services to other agents. The AMS provides agent lifecycle management for the platform. The ACC supports interoperability both within and across different platforms. The Internal Message Transport Protocols (MTPs) provides a message routing service for agents on a particular platform which must be reliable, orderly and adhere to the requirements specified by FIPA Specification XC00067- Agent Message Transport Service Specification. [34]

The ACC, AMS, Internal MTPs and DF form what will be termed the Agent Platform (AP). These are mandatory, normative components of the model. For further information on the FIPA Agent Platform see FIPA XC00023 - Agent Management Specification. In addition to the mandatory components of the FIPA Reference Model, the FIPA-OS distribution includes an Agent Shell, an empty template for an agent. Multiple agents can be produced from this template, which can then communicate with each other using the FIPA-OS facilities. [34]

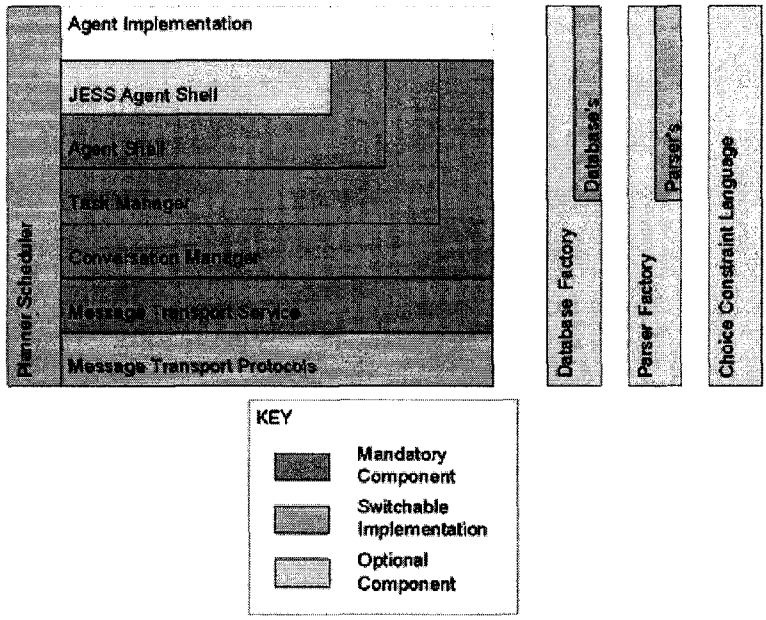


Figure 27 FIPA-OS Components [34]

The available FIPA-OS components and their relationship with each other are shown in Figure 27.

In first place we selected FIPA-OS as the agent development platform because it is very well documented, it has several tutorials and it has much more features. It was straightforward to install and run the platform, which offered us a graphical interface. However, during our experimentation we had some difficulties to run an agent that we implemented. To run our own agent was easier with JADE platform than FIPA-OS.

2.1.3.2.2 JADE

JADE [47] is a middleware enabling rapid development of multi-agent systems. It is composed of following elements

- A *runtime environment* where JADE agents can "live" and that must be active on a given host before one or more agents can be executed on that host.

- A *library* of classes that programmers have to/can use to develop their agents
- A *suite of graphical* tools that allows administrating and monitoring the activity of running agents.

Each running instance of the JADE runtime environment is called *Container* and it can contain several agents. The set of active containers form the *Platform*. A single special *Main Container* must always be active in the platform and all others containers register with it as soon as they start. The platform and containers relationships are illustrated in Figure 28.

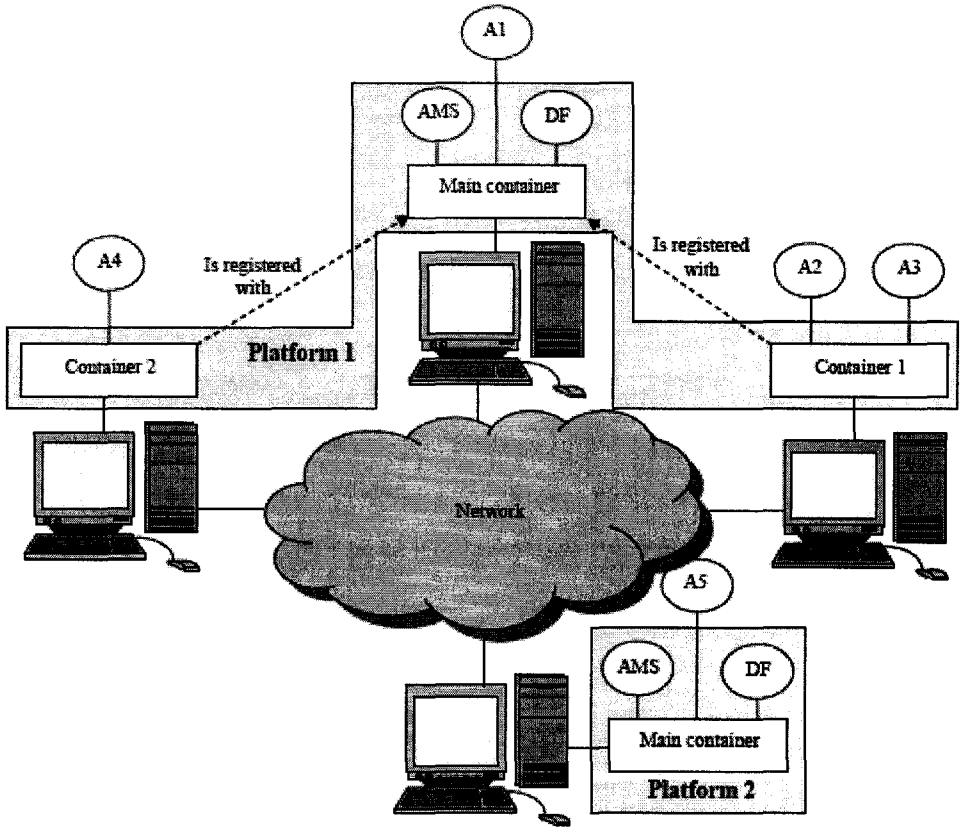


Figure 28 JADE Platforms and Container [47]

As a developer we don't need to know how the JADE runtime environment works, but just need to start it before executing our agents.

As shown in Figure 28, besides the ability to accept registrations from other containers, a main container has two special agents that normal containers don't have. Those two special agents are started automatically when the main container is launched. Those special types of agents are *AMS* (Agent Management System) and *DF* (Directory Facilitator).

The AMS provides the naming services (i.e. ensures that each agent in the platform has a unique name) and represents the authority in the platform (for instance it is possible to create/kill agents on remote containers by requesting that to the AMS).

The DF provides a Yellow Pages service that an agent can use to find other agents providing the services he requires in order to achieve his goals.

In our evaluation we witnessed and experimented how easy it was to create and run agents on this platform compared to other two platforms (FIPA-OS and ZEUS).

2.1.4 Database

For database issues, instead of designing our system for a specific database, Data Access Object (DAO) [26] structure as shown in Figure 29 should be used.

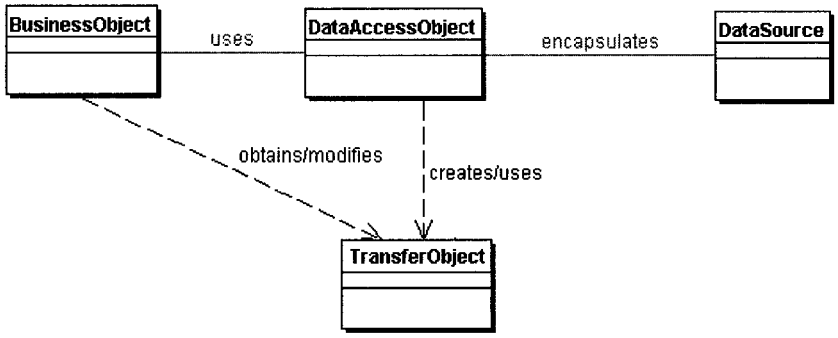


Figure 29 Data Access Object [26]

Therefore, all access to the data source will be abstracted and encapsulated. The access mechanism required to work with the data source will completely hide data source implementation details, enables transparency and easier migration, and will reduce code complexity in our system. Further details can be found in [26].

2.1.5 Standards Compliance

Data Crawler System should support OMG's CWM specification chapter 12 – Data Mining, JSR 73: JDM v1.0 and FIPA specifications.

2.2 Other non-functional specifications

DCS should be realized using open source product and legacy system (e.g. JADE, WEKA, etc.) because of the academic nature of the project.

Flexibility and modifiability are primary attributes of our system. Considering that the data crawler system is an academic project that will evolve and be improved eventually, new components will be added or existing components will change. As we selected descriptive DM approaches, the DM methods will be changed or adapted in order to perk up the quality of the produced knowledge.

In our project, computing time and responsiveness of the system is not an issue since we are working on a data mart; the designed system will not be online. Here online means that the system needs to classify quickly in order to accomplish the right action. (e.g. a system recognizes face of passing people in real-time to detect criminals in an airplane). Performance will be considered during the design but it will not be the primary attribute of our system.

3. ARCHITECTURE

The data crawler system conception is started based on the article [2], where the data mining process is proposed as an automated service. An automated data mining system is delivered by automating the operational aspects of the data mining process and by focusing on the specific client domains. The first criterion is met by using the autonomous agent to accomplish the data mining and JDM compliant data mining engine (DME). The second criterion is met by adapting the data mining process to the client domains as explained during the domain requirement description in section 4.1.

As a result, the system architecture is shaped by the agent technology and JDM proposed architecture.

3.1 Generic agent-based system architecture

The data crawler architecture is based on the generic agent application architecture proposed in [11], as shown in Figure 30.

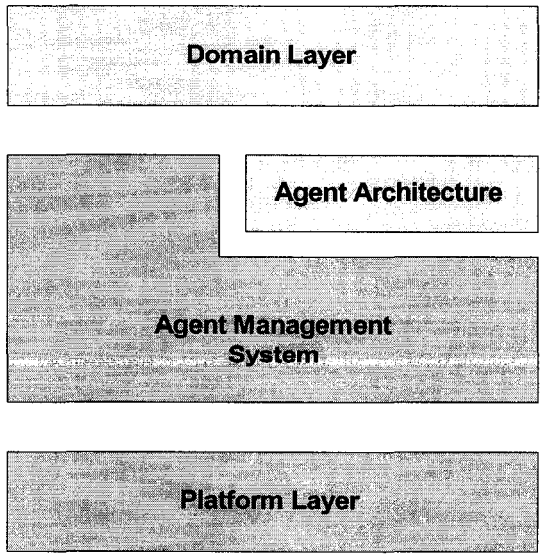


Figure 30 A Generic Agent-based System Architecture

In Figure 30, the Platform Layer corresponds to the hosts where the target system will run. The Agent Management System provides an interface for our agent to access the platform and the Agent Architecture implements the runtime environment. The Domain Layer relates the domain specific aspects.

Typically, Agent Management System and the Platform Layer are generic component that can be obtained from existing vendors or providers. The Agent Architecture and the Domain Layer are more specific to our problem therefore those are implemented.

3.2 JDM Architecture

JDM proposes architecture with three logical components: Application Programming Interface (API), Data Mining Engine (DME) and Mining Object Repository (MOR).
The API is an abstraction over the DME that provides the data mining services. The API let access to the DME. Therefore, an application developer that wants to use a specific JDM implementation need only to know the API library.

DME holds all the data mining services that are provided.

MOR contains all the mining objects produced by DME. It is used by the DME to persist data mining objects. JSR-73 don't impose a particular representation, therefore MOR could be in a file-based or in relation database form.

With this proposed architecture, we can easily change or upgrade our data mining implementation of our system as long as we are in agreement with JDM API.

3.3 Data Crawler Architecture

The data crawler architecture is based on the generic agent-based system architecture and JDM architecture as show in Figure 31.

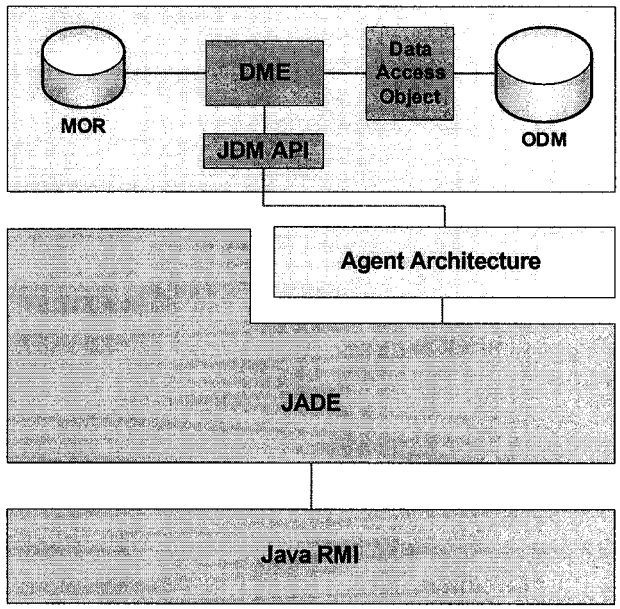


Figure 31 The Data Crawler Architecture

JADE is selected as the Agent Management System, which will run on Java RMI. Considering that the data mining is our domain and the data mining aspect of our system is implemented using JSR-073: JDM, then Domain Layer should be implemented using the data mining architecture proposed in [14].

4. SYSTEM REQUIREMENTS MODEL

This model will describe the system requirements in terms of agency and purpose and it involves the following phases:

- Domain Requirements Description
- Agent Identification
- Roles Identification
- Task Specification

4.1 Domain Requirements Description

In this step, common UML use-case diagrams are used to provide a functional description of the system. The following context diagram illustrates the actors interacting with the system.

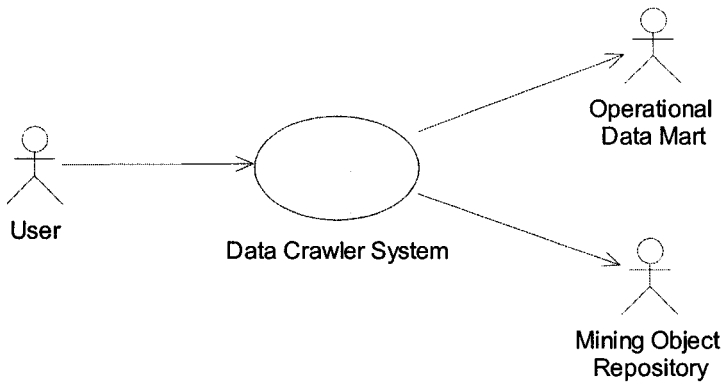


Figure 32 Context Diagram

Essentially, the Data Crawler System (DCS) has three actors: user, Operational Data Mart (ODM) and Mining Object Repository (MOR). User is the data mining clients; people from Bell Business Intelligence and Simulation teams or managers who are in

decisional context. User should access DCS to visualize produced models and to request for specific data mining tasks (e.g. applying a model produced by the DCS on data set specified by the user). *Operation Data Mart (ODM)* is the database where all data to be mined are picked up. The DCS should load data from ODM and should apply its data mining technique on this data and should save the results (e.g. models, statistics, apply results, etc.) and other mining objects (e.g. build tasks, build settings, algorithm setting, etc.) to the *Mining Object Repository (MOR)*.

The entire detailed functional description of DCS is presented in Figure 34. The domain description is based on the data mining process as shown in Figure 33.

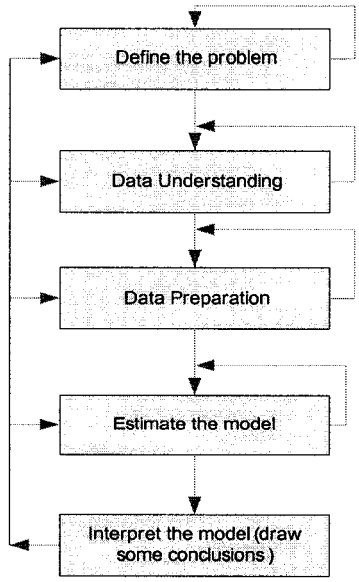


Figure 33 Data mining process

The first step "Define the problem" is about stating the problem and establishing the objectives of the project. It doesn't have any functional requirements; consequently there is no use case in Figure 34 related to this step. Plus, the last step "Interpret the model" is accomplished by the data mining clients and the functionalities related to this step is about presenting the results only.

In Figure 34, other functionalities such as "initialize system" or "manage data mining process" are also present. They are not related to the data mining process but to the system itself.

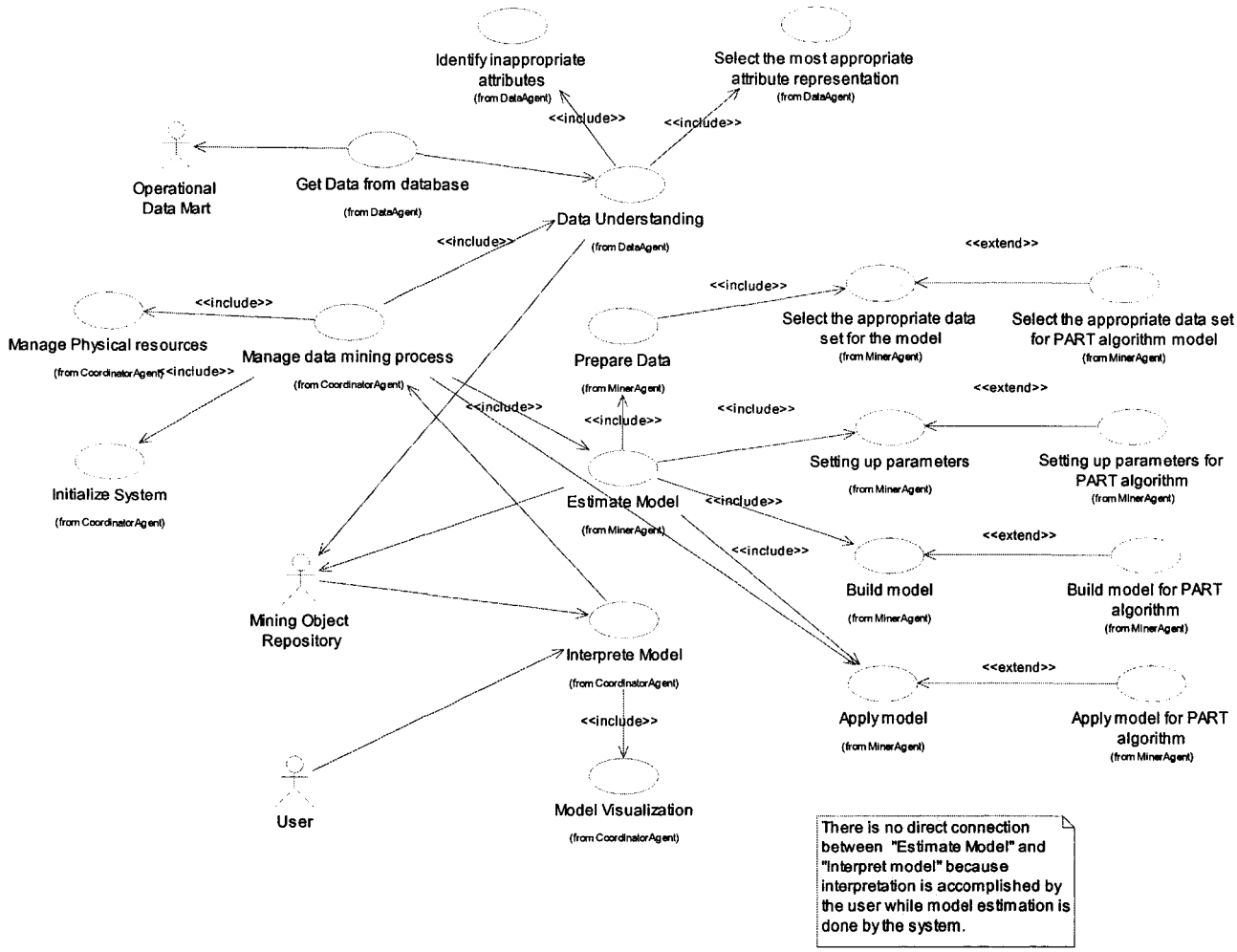


Figure 34 Domain Description Diagram

The rest of this section contains detailed analysis of each step of the data mining process shown in Figure 33. The domain requirements that are in the description diagram (Figure 34) are detailed in the following subsections.

4.1.1 Define the problem

In this initial step a meaningful problem statement and the objectives of the project are established. This step doesn't have any functional aspect, therefore it will not have any functional implementation of this step but this first phase is important to have a global understanding of the project and to establish non-functional requirement of the system.

As shown in Figure 35, the input to DCS is data obtained from Operational Data Mart. ODM has high dimensionality (it has more than 300 hundred variables "features") and it is a huge data repository (more than 1000 instance each day). The output depends on the DM method that is employed by the system; if we use a clustering method the output will be clusters, if the employed method is a decision tree technique the output will be a decision tree.

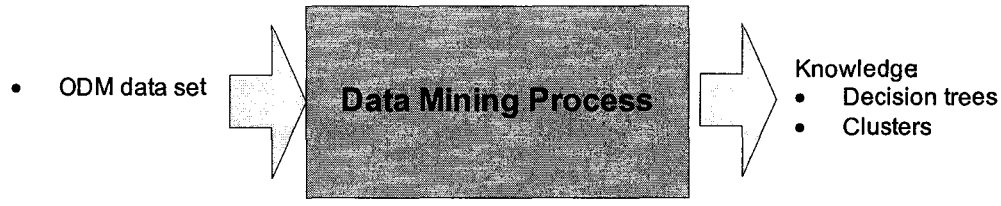


Figure 35 The data mining process as a black box

The main objective of the DCS is to extract unknown knowledge, possible interrelations and causal effects of variables in the database. There is no a priori hypothesis on how KD should be conducted; as a result descriptive data mining is the chosen approach. The data mining system should produce descriptive models. Autonomy is the principal characteristic of our system.

Systematically, DM system is supplied with data from ODM. The data supplying mechanism should be part of the system. Once the data is received, the DM system should perform each step of the data mining process shown in Figure 33. After completing of all these steps, the results and other mining object produced by the system are saved in MOR for future consultation by the users and next iteration should begin with a new data set.

The set of data mining methods should be predetermined in order to facilitate the automation of the DM process. We limit our self to unsupervised methods (descriptive methods) efficient in high dimensional space because of the high number of "feature" the input data has. PART algorithm may perhaps be a very good candidate.

Another aspect worth to mention is security. All operational data is about clients of Bell Canada and ODM contains sensible information about clients, thus confidentiality could an important point. DM system should extract and treat data autonomously without human interference and the data access should be accomplished from inside of Bell Canada. Therefore, security and privacy should not be a concern for us in this project because sensible data stays inside Bell Canada all the time.

4.1.2 Data understanding

The data understanding step is shown in Figure 36.

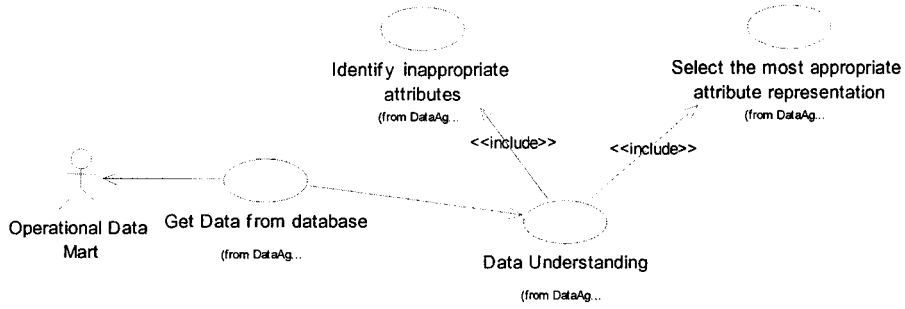


Figure 36 Data Understanding step diagram

Before explaining in detail the data understanding, the "Get Data from database" use case will be detailed. This use case represents the activity of getting data to be mined from a data source most commonly a database. As mentioned during the "define the problem" step, this activity should be part of the system too.

Data understanding is about data extraction, data quality measurement and detection of interesting data subsets. This task should be accomplished using exploratory data analysis (EDA) proposed in [4]. EDA is only applicable to numerical (float and integer) type attributes and categorical type attributes. EDA is a four step process as shown in Figure 37:

- Identifying inappropriate and suspicious attributes
- Selecting the most appropriate attribute representation
- Creating derived attributes
- Choosing an optimal subset of attributes

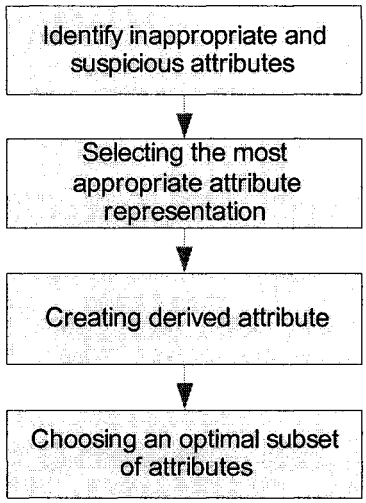


Figure 37 EDA process steps

The EDA method can not be used in our project as it is proposed in [4]. First of all, this method aims to automate predictive data mining process and in our project we are doing descriptive data mining which is very different. Considering that we have no prior directive on what we are searching with descriptive data mining, we can't tolerate any assumptions on data.

Also, as recognized in [4], by reducing the number of attributes and transforming data representation, we will not only contribute in computation time and memory requirement reductions and/or easily understandable models we will also cause some fatal loss of details contained in original data set. For that reason, the authors are proposing three association measures to try to balance the loss of details against the advantages of efficient data mining.

In descriptive data mining we can not allow any loss of information. Therefore, in our design we are not encouraging the pruning occurring during the data understanding, but in some evident case we can use it where there isn't any loss of information; the loss of information will mostly occurs with suspicious attribute.

However, the EDA process has great potential for the data crawler system even with its incongruity to our type of data mining. Therefore, it will be used as a starting point in designing the data understanding step by attuning it to the client context as shown in Figure 38.

- Identify inappropriate attributes
- Selecting the most appropriate attribute representation

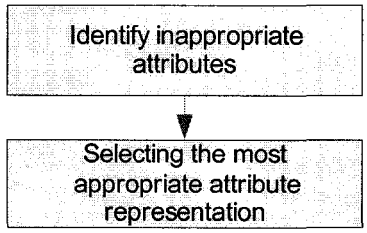


Figure 38 Data Understanding Process

In the following subsections, all steps of the EDA process as proposed in [4] are described in order to elucidate why some of them are not implemented and for future improvement of the data crawler system.

4.1.2.1 Identifying inappropriate and suspicious attributes

During the data understanding all inappropriate attributes are removed. Inappropriate attributes are described in Table IV [4]:

Table IV

Inappropriate attributes

Type	Description
Constant	Only contains a single value
Null	Has all missing values
Near Null	Has a fraction of missing values larger than a specified threshold
Many Values	Has a fraction or number of different values larger than a specified threshold.

More studies are necessary to establish a high-quality threshold for "near null" and "many values". At the moment in our project, for "near null" if missing values are larger than 98% (which is an arbitrary choice) the attribute will be considered as a "near null" attribute and for "many values" the specified threshold in 100% which means key attributes are rejected.

The suspicious attributes are described in Table V [4]:

Table V

Suspicious attributes

Type	Description
Artifact	Association and correlation with the target is greater than a specified threshold. These attributes are often unintentionally included and, without proper identification, they lead to artificially good models that do not generalize well.
Poor Predictor	Association and correlation with the target is less than a specified threshold. These attributes do not contribute much by themselves in predicting the target but combinations of these attributes may have increased predictive power. Attribute selection is the ultimate way to decide whether they are appropriate for further modeling
Near Constant	One values covers more than a specified fraction of all values
Few Values	Have less than a specified number of distinct values.
Few Cases	Have less than a specified number of distinct non-null cases.

The target attributes in the description of the first two types of suspicious attributes are the selected attributes and the sources attributes are the original attributes sets. Removal

of some attributes can cause serious loss of details contained in the original attributes, therefore we need to find a balance between this loss of information and efficiency of the data mining methods. To select the best set of attributes, the loss of details can be evaluated using association measures. Since the goal is to automate this process, we would like to have generic measures of association between the source and the target attributes. In this purpose, three types of association measures are proposed in [4]: mutual information, chi-squared Cramer's V and Goodman-Kruskal index.

For the rest of suspicious attributes, further analysis with client (Bell Canada) domain knowledge is necessary to determine the thresholds for identifying suspicious attributes. In [4], authors are talking about narrowing potentially hundreds of thousands of attributes down to a manageable subset, which is far from being our case. Also, as mentioned above, these association measures are to minimize the loss of information and will not to eliminate it. As a result, the DM system should not take care of suspicious attributes.

But a module and a user interface should be foreseen in the design of the system for future implementation to allow control over suspicious attributes if there is any change in our data mining approach.

```
for all attributes until all attributes are passed through
       Select the current_attribute;
       if current_attribute contains only a single value
              Set current_attribute as "CONSTANT";

       else if current_attribute has all missing value
              Set current_attribute as "NULL"


       else if 98% of current_attribute's instance are missing value
              Set current_attribute as "NEAR NULL"


       else if 100% of current_attribute's instance has different
       value
              Set current_attribute as "MANY VALUE"
Discard all attributes set as "MANY VALUE"
```

Algorithm 2 Identifying inappropriate attributes

Normally we should discard all attributes set as "CONSTANT", "NULL", "NEAR NULL" and "MANY VALUE" but they won't considering that we do the data mining incrementally (i.e. each data mining technique is applied on a small part of the whole dataset and we continue the data mining process using the same model produced earlier with another data sample and we continue this way until all dataset mined). But our data set is considered as infinite because the data (operational data from ODM) is growing every day as long as the company exists. Therefore, we can not conclude that the whole dataset is represented by a small part of it. For example, if the current data sample has an attribute as CONSTANT, we can not conclude that this attribute is CONSTANT for whole dataset because it could be the case for the current data sample only.

The "MANY VALUE" type attribute should be discarded because it represents the "key" attributes.

4.1.2.2 Selecting the most appropriate attribute representation

After identifying and rejecting inappropriate attributes, the retained attributes are processed to determine the most suitable representation. Outliers, missing values and encoding are handled during this step. In [4], for encoding, authors are suggesting discretizing for the numerical attributes by thresholding the original values into a small number of value ranges and for the categorical attributes, they are suggesting to merge together numerous categories similar to options in C4.5 [35]. The association measures, mentioned in the precedent section, are used to determine the optimal encoding.

During our examination of the data sample from Bell's ODM, we observed many missing values and outliers. For example, the categorical attribute FLAG_MET, where each instance has a value from the set [N, Y] (N for no and Y for yes), has instances with value of '9' which is probably a typo. In our implementation, no encoding can be performed because it can cause major loss of details present inside the original attributes. Considering the type of data mining strategy (descriptive) that we selected, even infinitesimal loss of information can not be tolerated. As a result, only outliers and missing values could be handled.

Also the data sample has many date type attribute. The date type will be considered as numerical attribute type because it doesn't have a standard representation, and must of the time it can be represented by a numerical value (e.g. in Java Date object is instantiated using a value of long type which represents the number of milliseconds that have passed since January 1, 1970 00:00:00.000 GMT).

The following subsections describes the strategy taken for outliers and missing values.

4.1.2.2.1 Outliers

The outliers are data samples that are significantly different or that do not comply with the remaining set of data.

For numerical attributes, we could use threshold values based on statistics to identify outliers. But in order to do that we need to know the data distribution, which isn't our case. Also, our data set is considered as infinite consequently we can't do any assumptions on data distribution. The outliers with the numerical attributes should not be handled.

For categorical attributes, a data sample that isn't in the group of possible values is considered as an outlier. A possible strategy will be to create a new category named "OUTLIER" each instance having an outlier in an attribute will be set to "OUTLIER" category. This approach can not be adopted because in some case, we could compute the distance between each instance and by adding a new category we need to specify the distance of the new "OUTLIER" category to other categories, which can distort the values. Also, some data mining technique can handle outliers and missing values so we don't need to do it at this step of the mining process. The outliers with the categorical attributes should not be handled.

4.1.2.2.2 Missing Values

The missing values are present within all type of attributes in ODM. Some data mining methods can't deal with them.

For numerical attributes, several strategies can be adopted. First, each missing value for a particular attribute can be replaced by a reasonable, expected, default value established following further analysis with the domain experts (Bell Canada) based on the domain

experience. We could also replace it by a mean value or a global constant value but by doing this the original data sample will be modified. In the initial version of the system no action is taken for the numerical type attributes. Moreover, some data mining algorithm supports missing values in dataset.

For categorical attributes, same strategy as with numerical type can be adopted; replace missing value by a possible default value based on the domain experience. For the same reasons as with the numerical attributes, this strategy can not be selected. In the initial version of the system no action is taken for the categorical type attributes.

No action should be taken for string type attributes because missing value can also be considered as a value.

```
for all attributes until all attributes are passed through
      Select the current_attribute;
      if current_attribute_type == numerical
            Do noting;

      else if current_attribute_type == categorical
            Do noting;

      else if current_attribute_type == string
            Do noting;
```

Algorithm 3 Selecting most appropriate representation

4.1.2.3 Creating derived attributes

The attributes derivation is for increasing the source attribute correlation with the target attribute. It is accomplished by using univariant transformation such as exponent, logarithm, quadratic function, inverse function, power function and square root. These transformations are typically only beneficial to linear regression models. Consequently, this step is performed only to continuous attributes otherwise it is deactivated. The Algorithm 4 can be used to derivate attributes

```
for all transformations (quadratic, inverse,power,square,exp,log)
until current transformation is accepted
      Compute correlation between source and target attributes;
      Apply transformation;
      Compute new_correlation between source and target attributes;
      if new_correlation > correlation
            Transformation accepted;
            Derived attribute kept;
      else
            Transformation rejected;
```

Algorithm 4 Transformation selection to create derived attributes

The concept of correlation can be generalized to continuous-categorical couple [5] so that these transforms can be expended to categorical attributes either.

In our project, the attribute derivation should be completely deactivated, considering that we don't have a viable reference dataset which has source and target attributes.

4.1.2.4 Choosing an optimal subset of attributes

Optimal subset of attributes selection without significantly affecting the overall quality of resultant model is for reducing computational time and memory requirements. Computational time and memory are not an issue to our project. Therefore, no more transformations should be applied on previously selected attributes. This will simplify our design too.

There are two algorithms suggested in [4] for attribute selection: *expectation of Kullback Leibler distance* (KL-distance) and *Inconsistency Rate* (IR). Those two algorithms can be used in future implementation if it is required.

4.1.3 Data preparation

Data preparation process is shown in Figure 39:

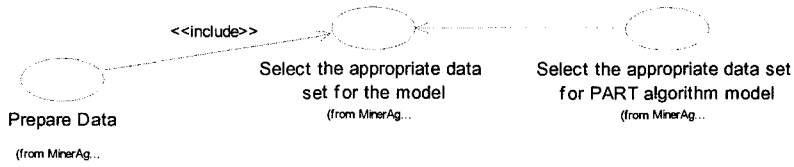


Figure 39 Data preparation step diagram

This step is about all activities to construct the final dataset for modeling. In other words, this step is for preparing the dataset selected during the data understanding phase for the DM methods used for modeling.

For example, clustering algorithms usually can only be applied to numerical or categorical data. Other type of data such as string need to be transformed to categorical data otherwise they are excluded. The general data preparation process is shown in Figure 40.

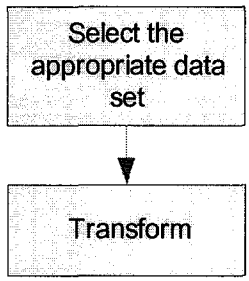


Figure 40 Data Preparation Process

Data preparation process is specific to a DM method. PART algorithm is selected to show how data preparation step works and how it can be implemented. In subsection

4.1.3.1, the specific data preparation required for PART data mining method is introduced.

4.1.3.1 PART Algorithm's data preparation

PART algorithm can only be applied to attributes where we can calculate the distance between each point such as numerical type attributes. Thus, string type variables should not be mined using Part algorithm. However, it can be applied to categorical if there is notion of distance between each category, otherwise, it can not.

For example, if we have a set of category such as [A, B, C, D, E] and the distance between each group is as shown in Table VI then the PART algorithm can be applied.

Table VI

Distance between each group

	A	B	C	D	E
A	0	1	2	3	4
B	1	0	1	2	3
C	2	1	0	1	2
D	3	2	1	0	1
E	4	3	2	1	0

PART algorithm can't be used, if there is no concept of distance in each category. Further analysis is required with client (Bell Canada) to determine which categorical attribute has distance concept.

For the moment, only numerical data should be mined using PART algorithm.

No further transformation on the selected attributes is required, since only numerical data are selected. Accordingly, the following simple algorithm (Algorithm 5) is used for data preparation for PART method.

```
Select all numerical type attributes.
Create a new data subset with the selected attributes.
```

Algorithm 5 Data preparation for PART algorithm.

4.1.4 Estimate the model

Estimate the model process is shown in Figure 41:

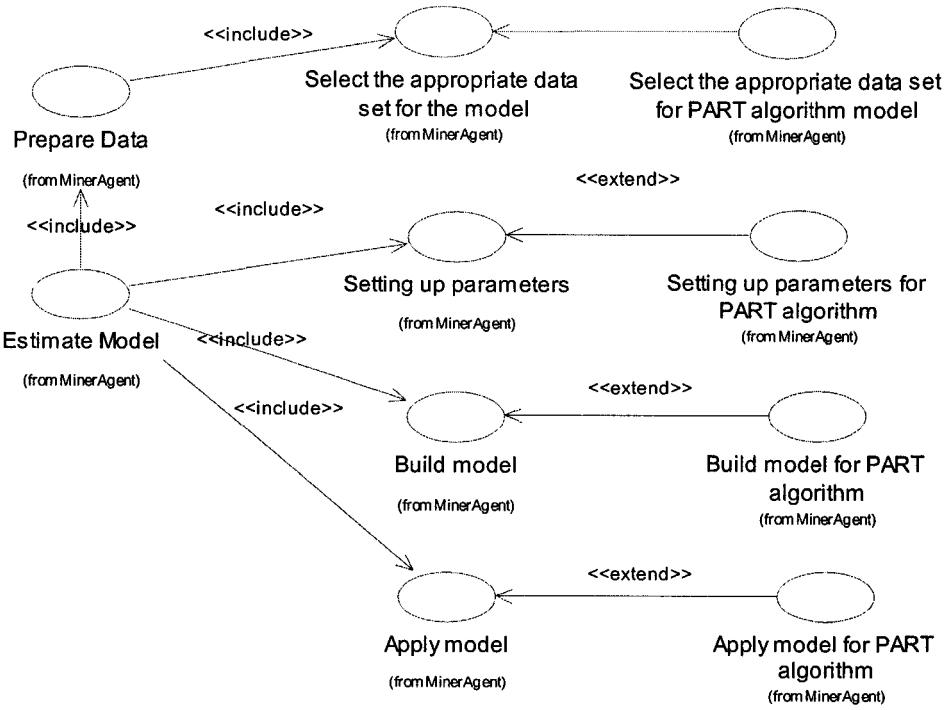


Figure 41 Estimate the model step diagram

During this phase, the DM algorithms are applied to the prepared dataset.

As shown in Figure 20, in our design the "prepare data" step is included in the "estimate data" step because each data preparation phase is related to a DM method. For example, if we are building a PART algorithm model we will necessarily prepare the data according to this specific algorithm.

In this section, first the general process flow of the model estimation is described. Then, a more precise description is given according to our project and the selected descriptive DM method.

4.1.4.1 General "Estimate the model" process flow

The modeling process (and all our DM system) is designed in agreement with Java Specification Request 73: Java Data Mining (JDM). The general modeling involves a four-step process as shown in Figure 42:

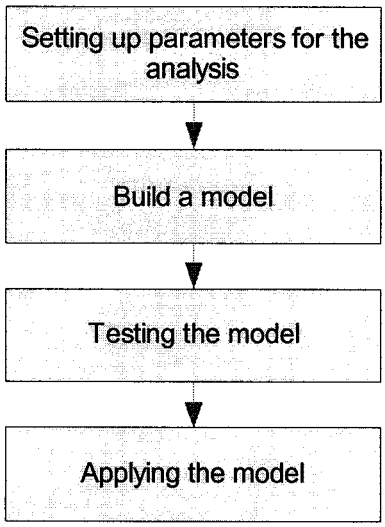
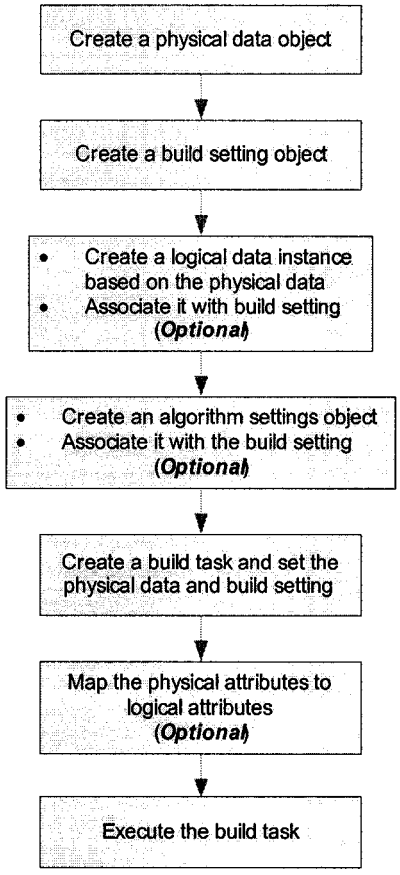


Figure 42 General "estimate the model" step

Detailed description of each step is below:

- *Setting up parameters for the analysis*: During this step all parameters or inputs that affect model building are set. The parameters or inputs values are predetermined. A detailed analysis is necessary to identify the parameters for each model.
- *Build a model*: A model is build. The model is a compressed representation of input data and it contains the essential knowledge extracted from data. The model object is specified by JDM. The scenario for model building is described in Algorithm 6:

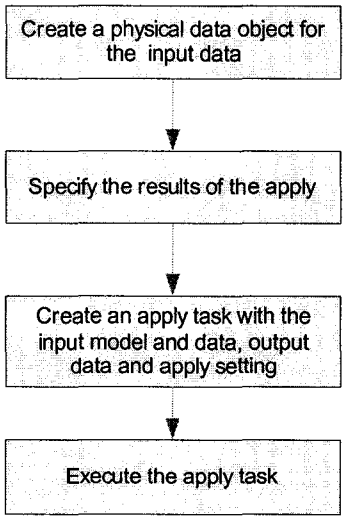


Algorithm 6 Model building scenario

- *Test the model*: Model testing estimates the accuracy of the model in predicting the target of a supervised model. It is processed after model building. The inputs are the model and a data sample.

  Considering that the data crawler system is designed to do descriptive data mining using unsupervised methods, the test task should not be implemented.

- *Apply the model*: Model applying is used to make predictions. For example, with clustering, during the "apply the model" step, a client can verify how well a data set fits within a given cluster and the data crawler system can give a probability indicating the accordance of the given data set to the selected cluster. The scenario for applying a model is described in Algorithm 7.



Algorithm 7 Apply building scenario

4.1.4.2 Specific "Estimate the model" process flow

The "estimate the model" phase should be implemented without the "test the model" step as shown in Figure 43.

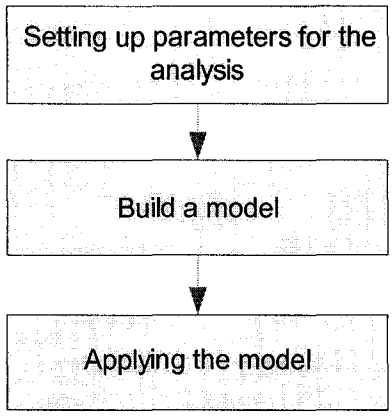


Figure 43 Specific "estimate the model" phase

"Test a model" step is removed from our model because it can only be applied to supervised methods (i.e. classification) and in DCS, we are only allowed to use unsupervised methods.

4.1.5 Interpret the model

Interpret the model process is shown in Figure 44:

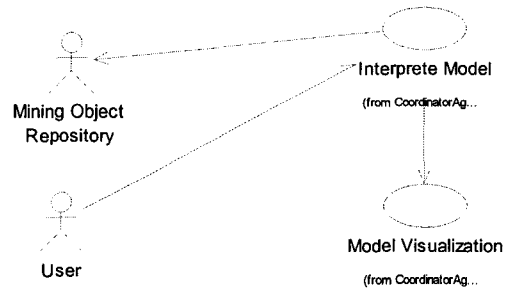


Figure 44 Interpret the model step diagram

This step will be accomplished mostly by the user of DM system. The problem of interpreting the resultant models is very important since the user does not want hundreds

of pages of numeric results. This kind of results is not understandable and/or interpretable and it will be difficult to make a successful decision.

This step should have two major functionalities:

- *Allow visualization of produced models.* The system should provide an interface for users to visualize the produced models.
- *Allow user to apply a model.* The system should allow user to verify the models that it is producing. Since, our main goal in creating this system is to automate the data mining process and help user in there data mining tasks, our system should inform the users when a specific situation (or event) happens. Thus, the user should be able to set some parameters and as a result the system should signal user when there is anomalies according to the set parameters.

All the functionalities above should be grouped in a graphical user interface (GUI) to facilitate the interaction of the user with the system. Another, functionality that isn't related to data mining but to the system itself is the ability to monitor the system resource. Therefore, the user should be able to monitor the available resources such as computing nodes, agents, in order to use them more proficiently. This functionality should not be implemented but it could be interesting to implement it in future releases.

JDM offers possibility to compute attributes statistics to interpret models appropriately. A task that computes statistics on physical data is provided by JDM. Therefore, this functionality could be implemented according to statistics API package using the *compute statistics task*, on a given physical data set.

Also, JDM 1.1 does not support data transformations and visualization functionalities; as a result the "interpret the model" step is not supported by JDM. Therefore, the current step will be implemented from scratch.

4.1.5.1 Model Visualization

The model visualization object is used to picture the content of a model. It should have a model detail object as input. Considering that representation of each model is different (specific to a particular algorithm), the visualization of each model also should be different accordingly to the algorithm used in the model. Initially, the model will be presented in a textual format; eventually graphical representation can be used.

4.1.6 System Related Functional Requirements

System related functional requirements are shown in Figure 45.

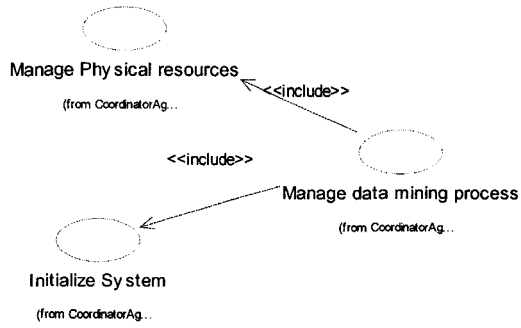


Figure 45 Manage data mining process diagram

There are three types of system related functional requirements:

- DM process management
- System initialization
- Resources management

All those functional requirements are described in the following subsections.

4.1.6.1 DM process Management

This use case is about keeping track of the mining process and others related activities such as system initialization, resources management and errors management.

The DCS should keep track of the data that has been mined and with which available DM methods. All produced models must be kept in a database (i.e. mining object repository "MOR") and made available to the user when requested. DCS must record when fault occurs during the lifetime of the system and if possible take necessary actions to recover or repair.

System initialization and resources management are described in the following sections.

4.1.6.2 System Initialization

During the system initialization, the following three tasks should be executed.

- *Load resources information*: physical resources (i.e. available host on which the agents can be executed for data mining)
- *Load Input data information*: all information related to accessing database and to tracking all the data mined and still to be mined
- *Load data mining info*: all information related to possible data mining algorithm implemented in the system and where the DME and MOR are located and how to access them.

The information can be stored in text files that are read during the initialization of the system. Those load files should be updated during the lifespan of the system, thus they could be used to restore the system states, reduce the lost of information if a failure occurs or simply if the system had to be stopped.

4.1.6.3 Resources Management

The resources management consists of deciding on which node which agent will be executing. The resources management should be done according to the following rule:

> *For each node, there should be only one Data agent and one Miner agent executing.*

There should be only one Coordinator agent on the system. At the start-up, each agent should be created according to the rule above. More analysis and studies are necessary to establish a more efficient resources management strategy.

4.2 Agent Identification

During the domain requirements description, the functional description of the system is provided without getting into any discussion about agents. In this step, the separation of responsibility concerns into agents is accomplished. This step is launched from the use case diagrams of the previous step where each use-case instance is associated with an agent. Each agent is represented by an UML package. Therefore, each package defines the functionalities of a specific agent.

The selection of the use cases that will be part of an agent is done following the criteria of functionality coherence and cohesion. For example, the data preparation activities of the data mining process are divided between two agents: data agent and miner agent. The Data agent realizes the data understanding step of the data mining process. It should extract data and select an interesting subset of data. This subset of data isn't yet bound with a specific data mining algorithm, while the data preparation step of the data mining process is specific to a data mining algorithm; thus this step should be accomplished by

the miner agent. Therefore, data understanding step should be associated to a different agent than the agent that realizes the data preparation and estimate model steps.

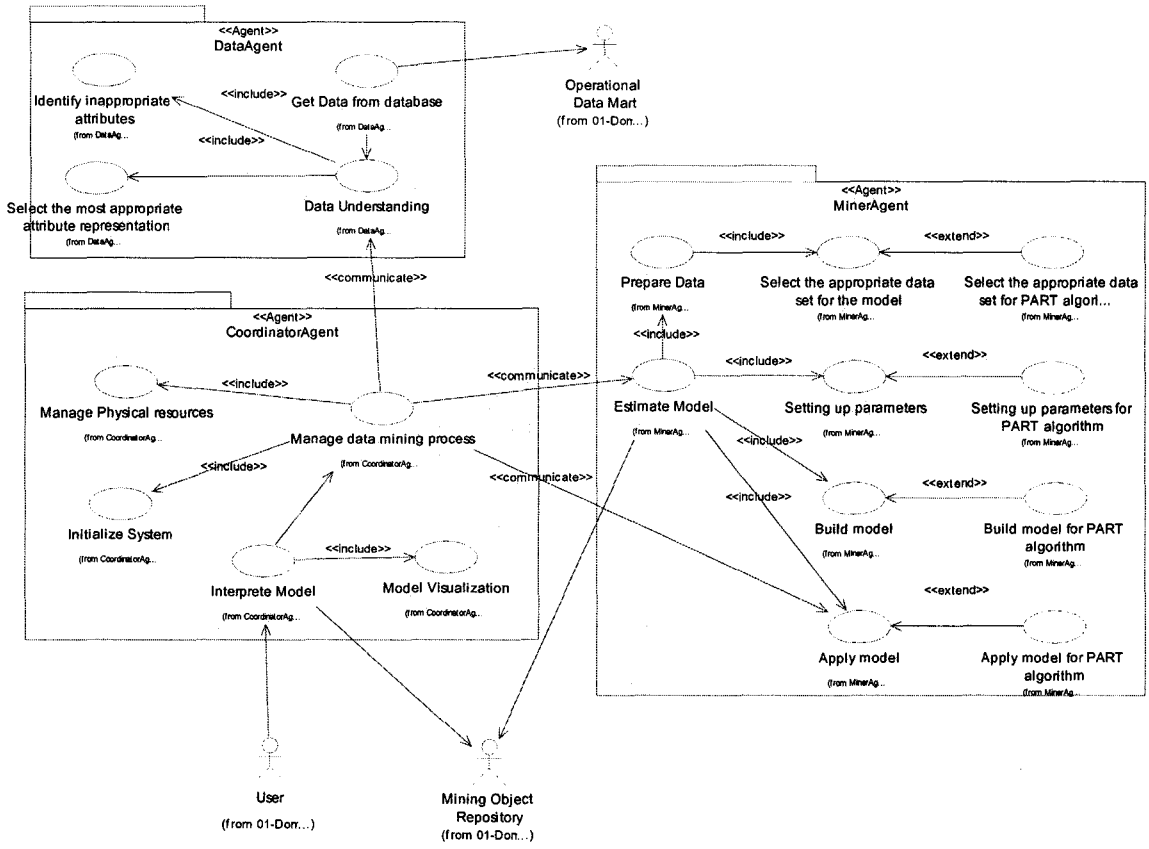


Figure 46 Agent identification diagram

Basically, three agents have been identified:

- Data agent,
- Miner agent and
- Coordinator agent.

Data agent acquires data from ODM database and accomplishes the "data understanding" step which consist of data extraction and data quality measurement and detection of interesting data subsets as described in section 4.1.2. Miner agent should prepare the previously cleansed data for the data mining algorithm and should produce a model that will be saved on MOR. This agent should also apply models on data following user's instructions.

Coordinator agent has mainly three tasks:

1. *Physical resources management:* In our context, the physical resources are the host on which the agents run. Managing activity consists of deciding on which node which agent will be executing. The resources management should be as follow:

For each node, there should be only one Data agent and one Miner agent executing.

   There should be only one coordinator agent on the system. In our implementation the physical resources management is implicit, which means that the coordinator agent will not take any particular actions to realize it. At the start-up, each agent should be created according to the rule above. The coordinator agent should activate or deactivate the data and miner agents depending on the available jobs. At this level the Physical resources management is too simple but when we will have more complicated mining strategy with limited resources then this separated functionality will be important.

2. *Data mining management:* The data mining management consists sequentially of
   - Requesting data agent to get and prepare data for mining,
   - Requesting miner agent to prepare and mine the cleansed data and
   - Keeping track of the data mining process within ODM's data. All data should be mined in sequence.

3. *Error management:* At this level it consists of logging the errors that occur.

4. *User interaction:* It should allow users to visualize resulting models as described in section 4.1.5 and apply models on user defined data set. Eventually, other type of task that facilitates the model interpretation could be integrated to the system.

4.3 Roles Identification

This step consists of exploring each agent's responsibilities through role specific scenarios using sequence diagrams. Basically, all possible paths of the Agent identification diagram involving inter-agent communication (path with <<*communicate*>> stereotype) will be explored. A path corresponds to a scenario of an interacting agent that will achieve a behavior of the system. Each agent can have more than one role and each object in the sequence diagram is represented with the syntax: <role name>:<agent name>.

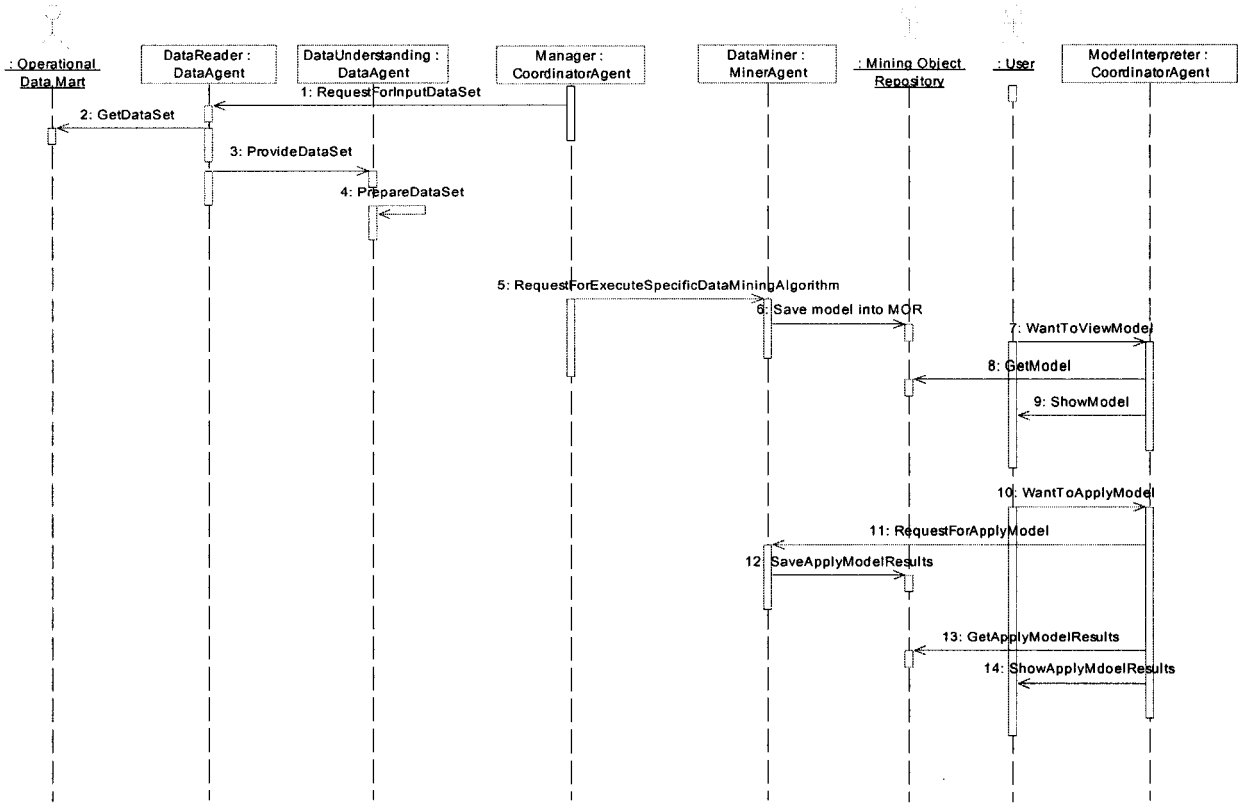


Figure 47 Role Identification diagram

Figure 47 shows the scenario of the data mining process. The scenario is described as follow:

- The *Manager* requests *DataReader* to get a data subset from ODM.
- Given the data information and access information, DataReader acquires data subset from ODM and provides it to *DataUnderstanding* which should cleanse the data.
- Once the data is cleansed, the Manager should ask the *DataMiner* to mine the cleansed data using a specific algorithm.

- Once the data mining process is finished, the *DataMiner* should save the resulting model into MOR and it should inform Manager of its current status. Then, the Manager should give the DataMiner another job and so on.
- The user asks the *ModelInterpreter* to view a model. Then the ModelInterpreter gets the model from MOR and shows it.
- The user asks the ModelInterpreter to apply a model on a specified data. Then the ModelInterpreter requests Manager for applying a model on a specified data.
- Given the model and data, the Manager request DataMiner to do apply model task on the data.
- Once the apply model task is finished, the DataMiner should save the apply results into MOR and should inform the Manager that the apply model task is completed and then the Manager informs the ModelInterpreter of the completion that of the apply model task.
- Once the apply model results are ready, the ModelInterpreter shows them.

4.4 Task Specification

In this section, the capabilities of each agent are specified using activity diagrams. The relationship between activities means messages and communications between tasks of the same agent. There are two columns in diagram, first one contains the tasks from other agents interacting with the agent and the second one contains the tasks of the agents describing its behavior.

The task specification of each agent is described in the following subsections.

4.4.1 Coordinator Agent

The task specification diagram of the Coordinator agent is shown in Figure 48.

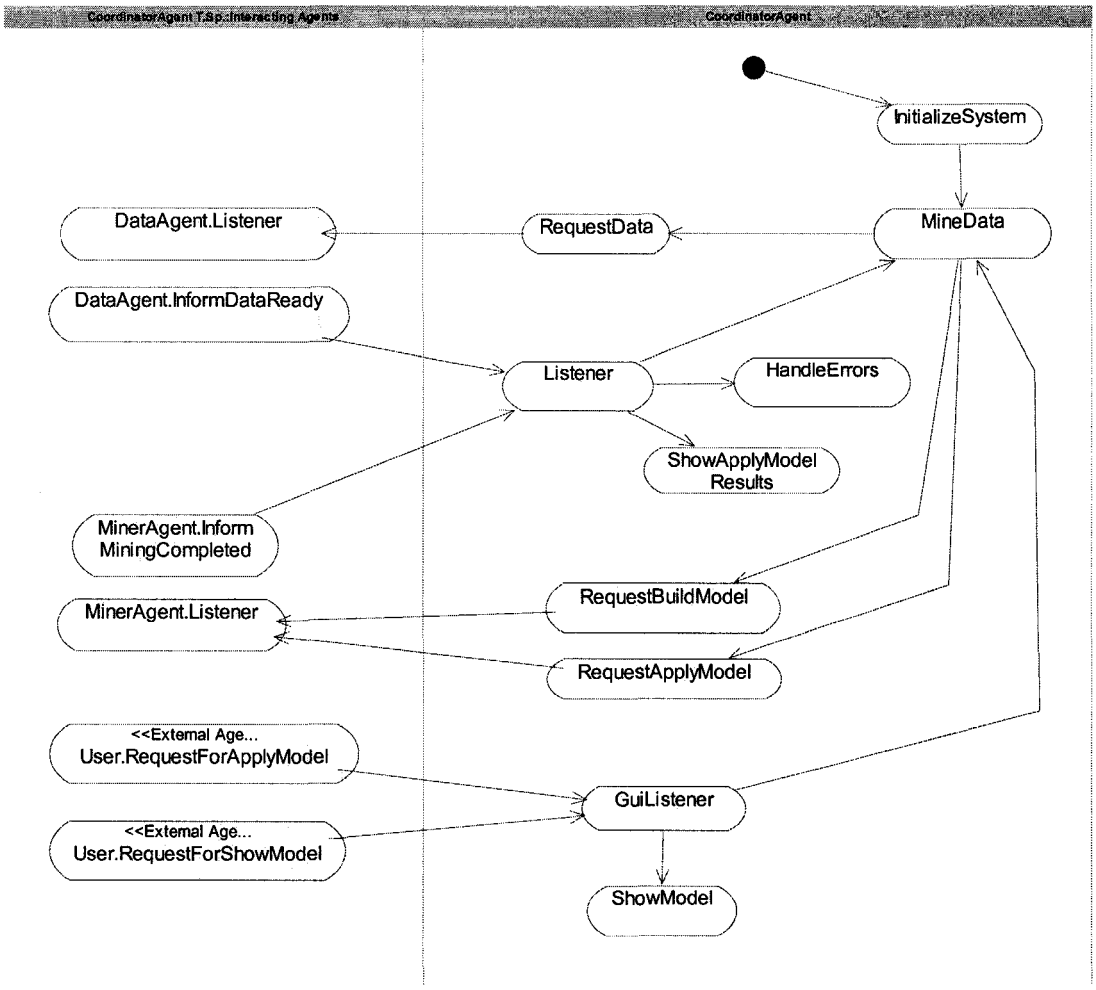


Figure 48 Task Specification Diagram of Coordinator Agent

The Coordinator agent is the one that initialize, start and manage the system. The Listener task is used to pass incoming communications from other agents to the proper task. MineData task is used to supervise the data mining process. The physical resources management also is accomplished within this task. RequestData, RequestApplyModel and RequestBuildModel tasks correspond to outgoing messages to DataReader and DataMiner of the Roles Identification diagram in Figure 47.

InformMiningCompleted task informs Coordinator agent that the mining task is completed. If the completed mining task is a build task then the Coordinator request Miner agent to realize another build task. If the completed mining task is an apply task then the Coordinator agent should call for ShowApplyModelResults task that will collect the apply results from MOR and show them in the user interface.

The Coordinator agent should also interacts with the user. GUIListener task dispatches the user requests to the proper tasks. User can do two requests: RequestForApplyModel and RequestForShowModel. When user requests for an apply model, the Coordinator agent call for MineData task that should call in his turn the RequestApplyModel task after deciding which Miner agent should do the apply task. Then the RequestApplyModel task should request Miner agent for applying a model on data both identified by the user. ShowModel task should show a model in the user interface in response to user request.

Finally, LogErrors task records all errors.

4.4.2 Data Agent

The task specification diagram of the Data agent is shown in Figure 49.

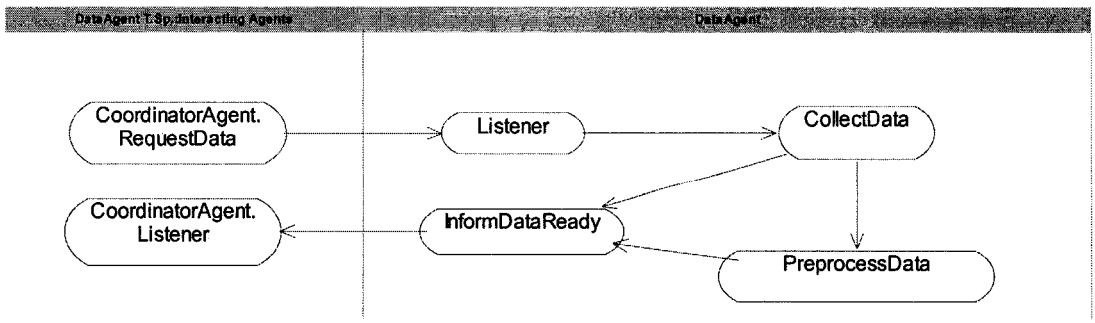


Figure 49 Task Specification Diagram of Data Agent

The Listener task should pass the incoming message from Coordinator agent to CollectData task, which will collect data from a data source and call PreprocessData task. The data source address and other information related to data to be collected are given by the Coordinator agent. If an error happens during the data collecting, InformDataReady task should be called and the Coordinator should be informed of the error.

The data understanding step of the data mining process is realized with the PreprocessData task. Thus, this task should be implemented according to section 4.1.2Then InformDataReady should inform Coordinator agent that the collecting and preprocessing of the data is completed.

4.4.3 Miner Agent

The task specification diagram of the Miner agent is shown in Figure 50.

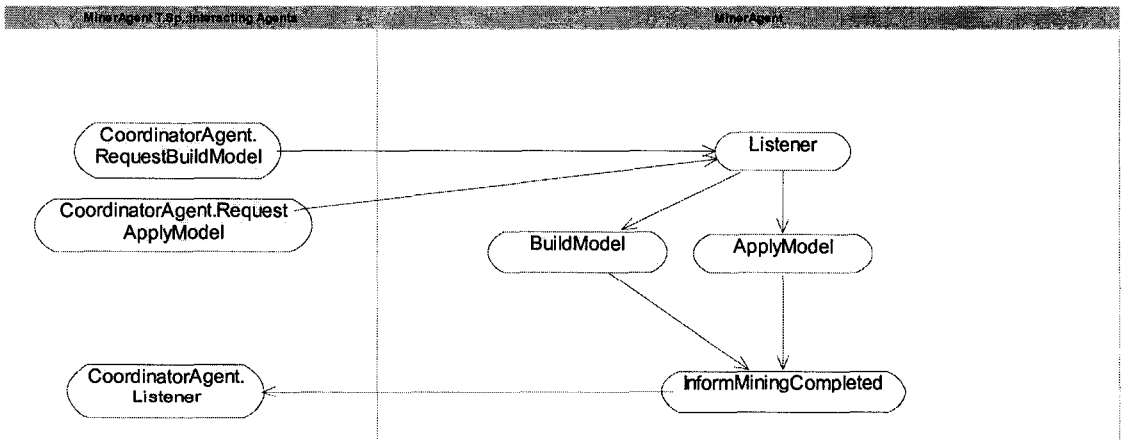


Figure 50 Task Specification Diagram of Miner Agent

Just like the Data agent, Miner agent too should process the incoming messages and call for proper task according to the Coordinator agent request. If build model is requested then BuildModel task should be scheduled for execution otherwise if apply a model is requested then ApplyModel task should be scheduled. Once the mining task (build or apply) is finished, the Coordinator agent should be informed. If an error occurs during the mining task, it should be communicated to the Coordinator agent too. BuildModel and ApplyModel tasks are implemented according to section 5.4

All the information related to the pruned input data and the setting parameters of the mining tasks (i.e. build setting, algorithm setting, clustering setting, etc.) are communicated by the Coordinator agent.

5. AGENT SOCIETY MODEL

This model will describe the social interactions and dependencies among the agents and it involves the following phases:

- Ontology Description
- Roles Description
- Protocols Description

5.1 Ontology Description

In this step the knowledge ascribed to individual agents and their communications are described using class diagrams and OCL (Object Constraint Language) constraints. The Ontology Description is composed of

- Domain Ontology Description and
- Communication Ontology Description

5.1.1 Domain Ontology Description

The domain ontology tries to outline the relevant entities and their relationships and rules within that domain using class diagrams. Thus, the agents can have the same understanding of their domain. The ontology is described in terms of

- concepts,
- predicates,
- actions and
- their relationships.

Concepts are categories, entities of the domain, *predicates* are assertions on the properties of concepts and *actions* are what are performed in the domains.

The domain ontology description diagram of the DCS is shown in Figure 51 below.

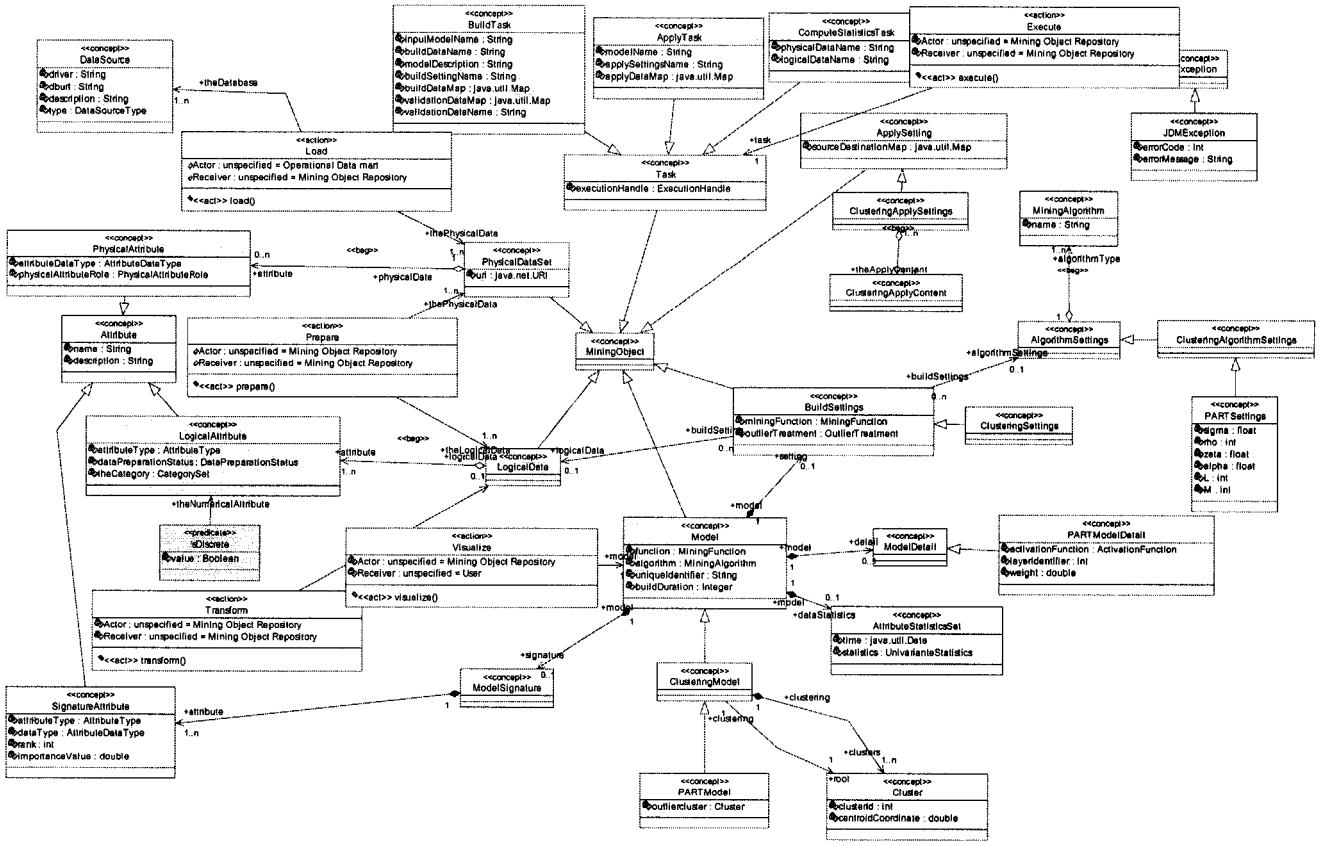


Figure 51 Domain Ontology Description Diagram

The following three standard UML relationships are used:

- *Generalization* permits the "generalize" relationship between two entities. For example, in Figure 51 the *BuildTask* is a generalization of *Task* concept.
- *Association* models a logical relationship between two entities and allows the specification of the role of the involved entities. For example, the *Execute* action should execute a *Task* concept.
- *Aggregation* is used to construct sets where their values can be restricted as specified in the W3C RDF specifications [43].

Our domain is data mining and all data mining aspect of our system is designed using JSR-73. Therefore, the domain ontology description is created using JDM specification. Some other aspects of DM described in JDM (such as test task, statistics, regression, classification, etc.) are not present in Figure 51, since they will not be used in this initial implementation and they will not complexify further our domain ontology description unnecessarily.

5.1.2 Communication Ontology Description

The communication ontology is about representing the social interactions between agents. The class diagrams are used to show all agents and all their interactions (lines connecting agents).

In the FIPA standards, a communication consists of speech acts [44] and is grouped in several interaction protocols that define the sequence of expected messages. A communication is drawn from an initiator to a participant and defined using three elements: protocol, language and ontology. Ontology is taken from the domain ontology description. Language should be Resource Description Framework (RDF) language [46] and the protocol should be FIPA standard protocols [45].

The communication ontology description of Data Crawler System is given in Figure 52.

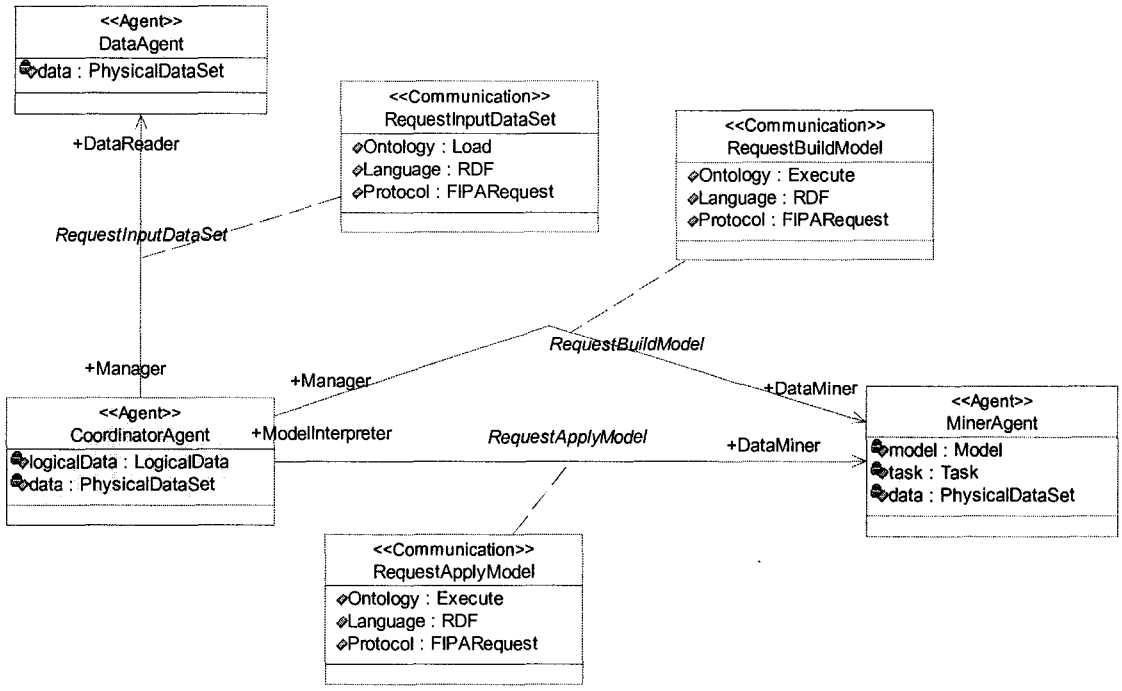


Figure 52 Communication Ontology Description Diagram

The attributes section of each class symbolizing an agent represents the knowledge of this agent and those knowledge elements are from Domain Ontology Description. For example, Data agent is aware of a PhysicalDataSet concept but doesn't know what a model or a task object is.

A common use with the "Request" communicative act is to request that the receiver performs another communicative act [45]. Therefore, all the communication should be in form of Request-Inform (where Coordinator will Request to Data agent to prepare data and will Request Miner agent to mine data, others will inform the result of their actions).

5.1.2.1 Communication between Coordinator and Data agents

Coordinator agent requests data agent to load a data set. The load action should have the URI (Unified Resource Identifier) of the input data and other information such as login name and password if required depending on the data source.

The "prepare" action also should be realized by the data agent even it is not explicitly requested by the Coordinator agent, since it make no sense to load a data set and not prune it.

The "inform" communicative act should contain the state of the data agent.

5.1.2.2 Communication between Coordinator agent and Miner agent

Coordinator agent requests miner agent to execute a specific task (apply or build task). In order to execute apply or build task, miner agent will need several information. The execute action should have the following information: the type of the task, the name of pruned data (the physical data set) that is stored in MOR and the setting parameters.

The "inform" communicative act should contain the state of the miner agent.

5.2 Roles Description

The Role Description step involves of showing the roles played by the agents, the tasks, the communication capabilities and inter-agent dependencies using class diagrams. All the rules that regulate the agent society and the domain in which the agent exists are introduced in this section.

The Role Description is shown in Figure 53. Each package represents an agent and each class represents a role that an agent plays. Each role is composed of several tasks that

define its behavior. In Figure 53, the tasks, that are composing a role, are located in the operation section of a role class diagram.

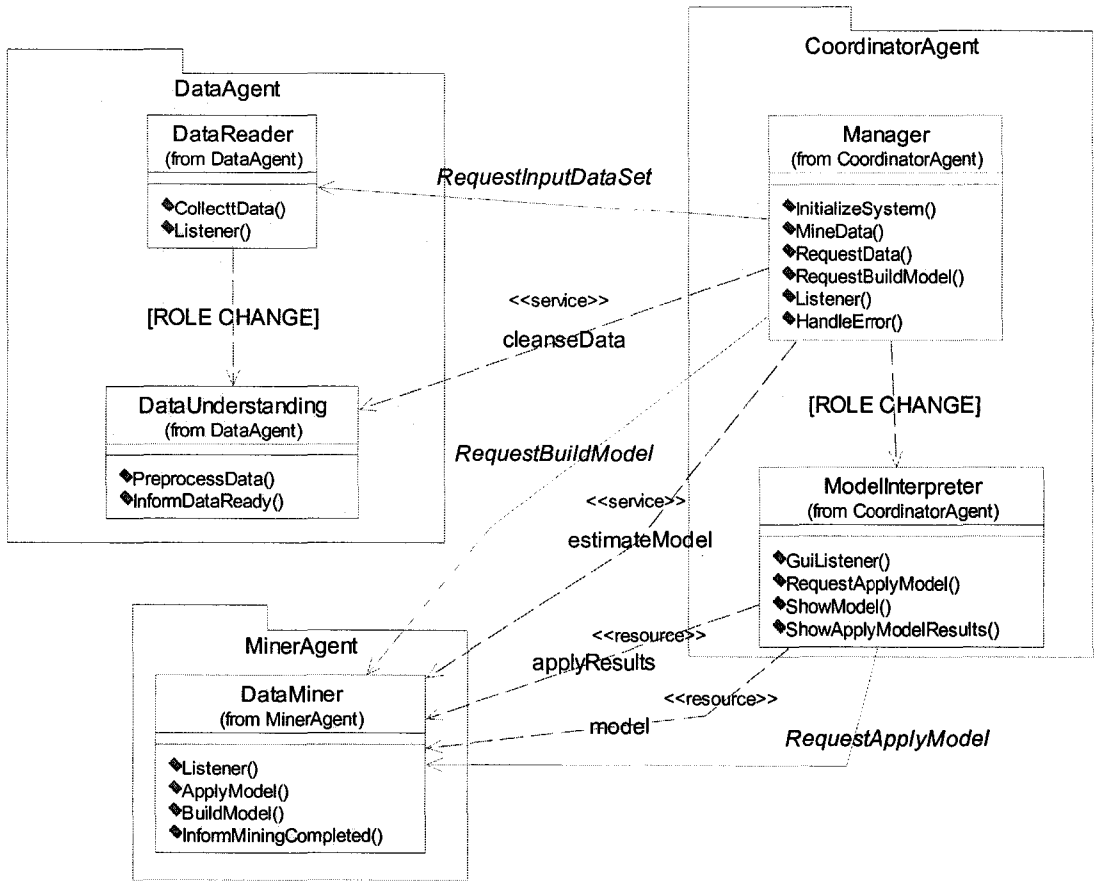


Figure 53 Roles Description Diagram

There are several relationships possible between agent's roles. The connections between roles of the same agent, using dashed line with the name [ROLE CHANGE], represent changes of role. It signifies the dependency between roles of a same agent. The second type of relationships specified by solid lines represents a conversation between roles of different agents. Those conversation relationships are the same as in the Communication Ontology Description diagram, therefore the same relationships name will be used to

keep a certain level of consistency (this consistency is ensured by the PASSI Toolkit). There is also two more type of relationships expressing the following type of dependency:

- Service dependency, represented by a dashed line with the <<service>> name, indicates that a role depends on another to bring about a goal
- Resource dependency, represented by a dashed line with the <<resource>> name, indicates that a role depends on another for the availability of an entity.

In Figure 53, coordinator agent has two roles: manager and model interpreter. The manager role is about controlling the data mining process by requesting some actions to be realized by other agents (data and miner agents) and taking some actions itself in response to other agents reply to the coordinator agent requests. The model interpreter role is about interacting with the user. If user want to see a model produced by the system then show it to user or if user want to apply a model (produced by the system) to a data set then ask a miner agent to apply the model. Once the apply task is finished by the miner agent then show the apply results to the user.

Data agent too has two roles: data reader and data understanding. Data reader role simply collects the data and the data understanding role will cleanse the raw data for miner agent to mine it eventually.

Miner agent has only one role which is data miner. This role is about preparing the cleansed data and building a model or applying a model on the prepared data.

The coordinator agent depends on data agent for the accomplishing the data understanding step of the data mining process which is represented by the service dependency cleanseData. Also, it depends on miner agent for the model estimation which is represented by the service dependency estimateModel. The model interpretation can be done only we have the data mining results such as models and/or

apply results, which are represented by the resource dependencies applyResults and model.

5.3 Protocols Description

Protocol Description uses sequence diagrams to specify the set of rules of each communication protocol based on speech-act performatives. In our case, FIPA standard protocols [45] are used. Therefore, we don't need to specify a custom protocol for DCS.

6. AGENT IMPLEMENTATION MODEL

This model describes the agent architecture in terms of classes and methods. There is two level of abstraction: multi-agent level and single-agent level. This model is composed of two phases as follow:

- Agent Structure Definition
- Agent Behavior Description

The structure of the agent-based system is described using conventional class diagrams and the behavior of the agents (multi-agent level and single-agent level) is described using activity diagrams and state diagrams.

6.1 Multi-Agent Structure Definition

The multi-agent structure definition (MASD) is shown in Figure 54. Each class symbolizes an agent identified during the Agent Identification phase and actors are reported to illustrate the agent's interactions with their environment. Attributes section of each class represents the knowledge (which are discussed during the communication ontology description) of the corresponding agent. The operations section illustrates the tasks of the agent.

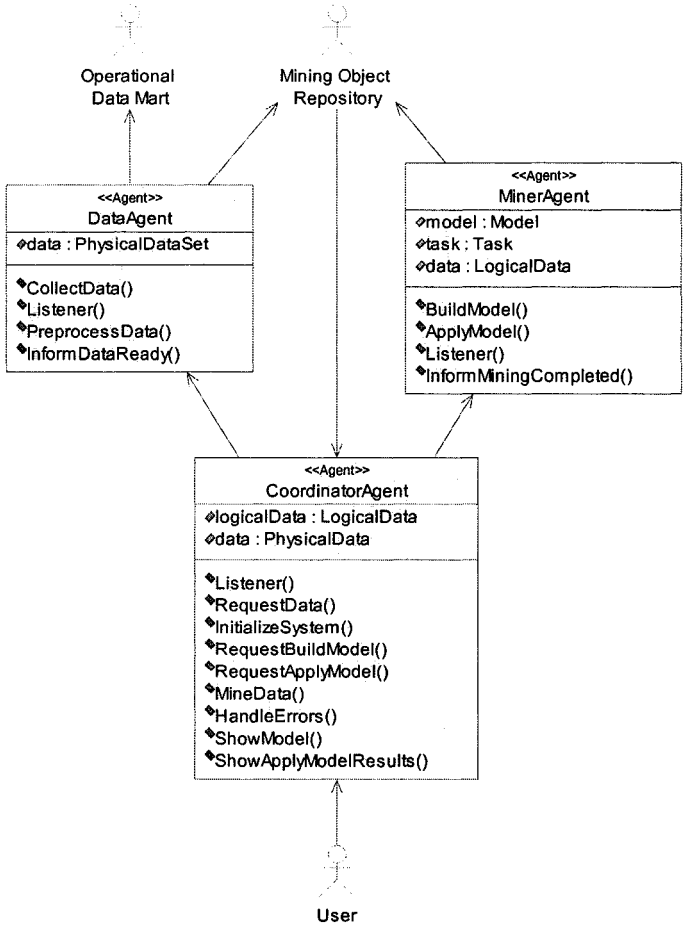


Figure 54 Multi-Agent Structure Definition diagram

For example, there is no direct interaction between data agent and miner agent and the coordinator agent communicates with both agents. Only the coordinator agent interacts with the user. All agents has access to MOR: data and miner agents write down mining objects that they produce such as pruned data, models, etc. and coordinator agent get those objects to show them to the user. Only data agent has access to ODM to get raw data.

6.2 Multi-Agent Behavior Description

Multi-agent behavior description of DCS (shown in Figure 55, Figure 56 and Figure 57) illustrates the whole system behavior. The diagram is cut into 3 figures because it didn't fit into one page.

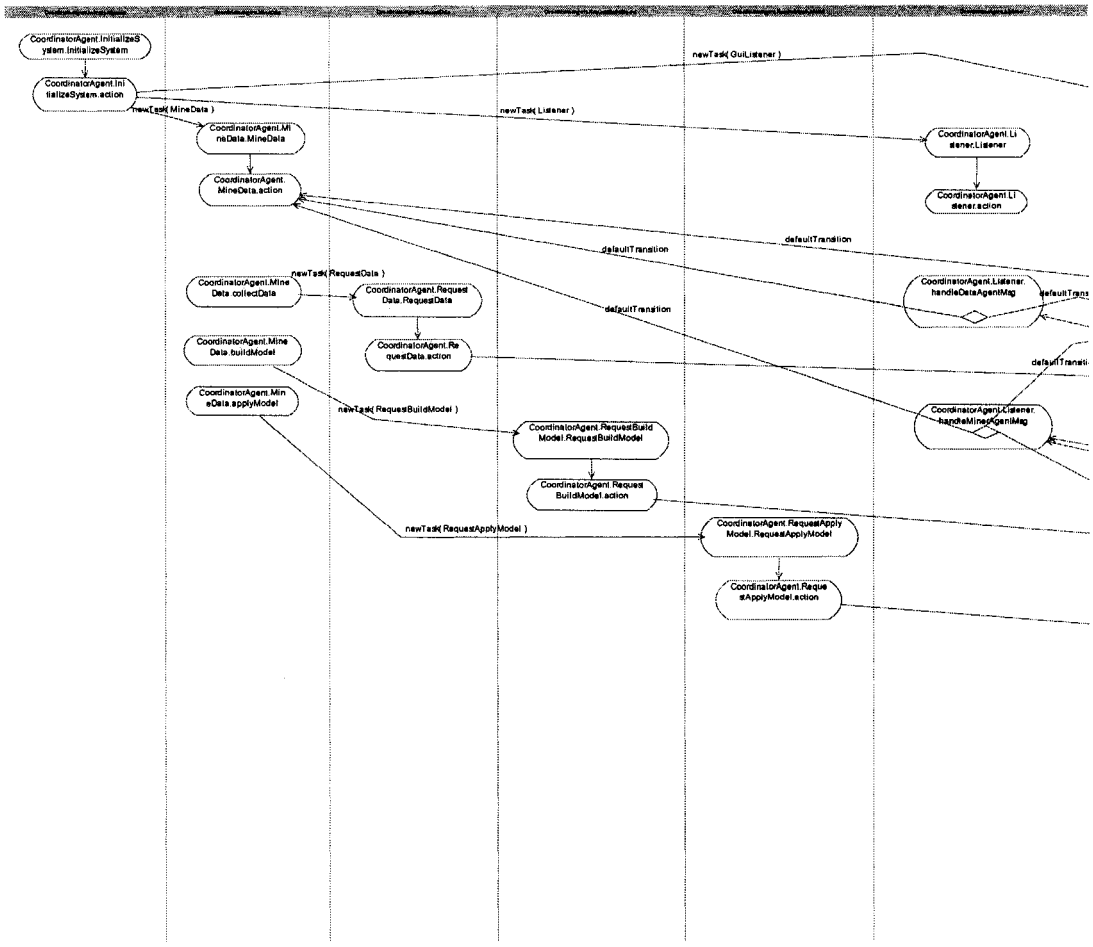


Figure 55 Multi-Agent Behavior Description Diagram

The activity diagram shows the flow of events among agent's tasks. Each swimlane (a column in the diagram) represents an agent's task class (i.e. <<agent>>.<<task>>). The activities inside the swimlanes represent the methods of the related class.

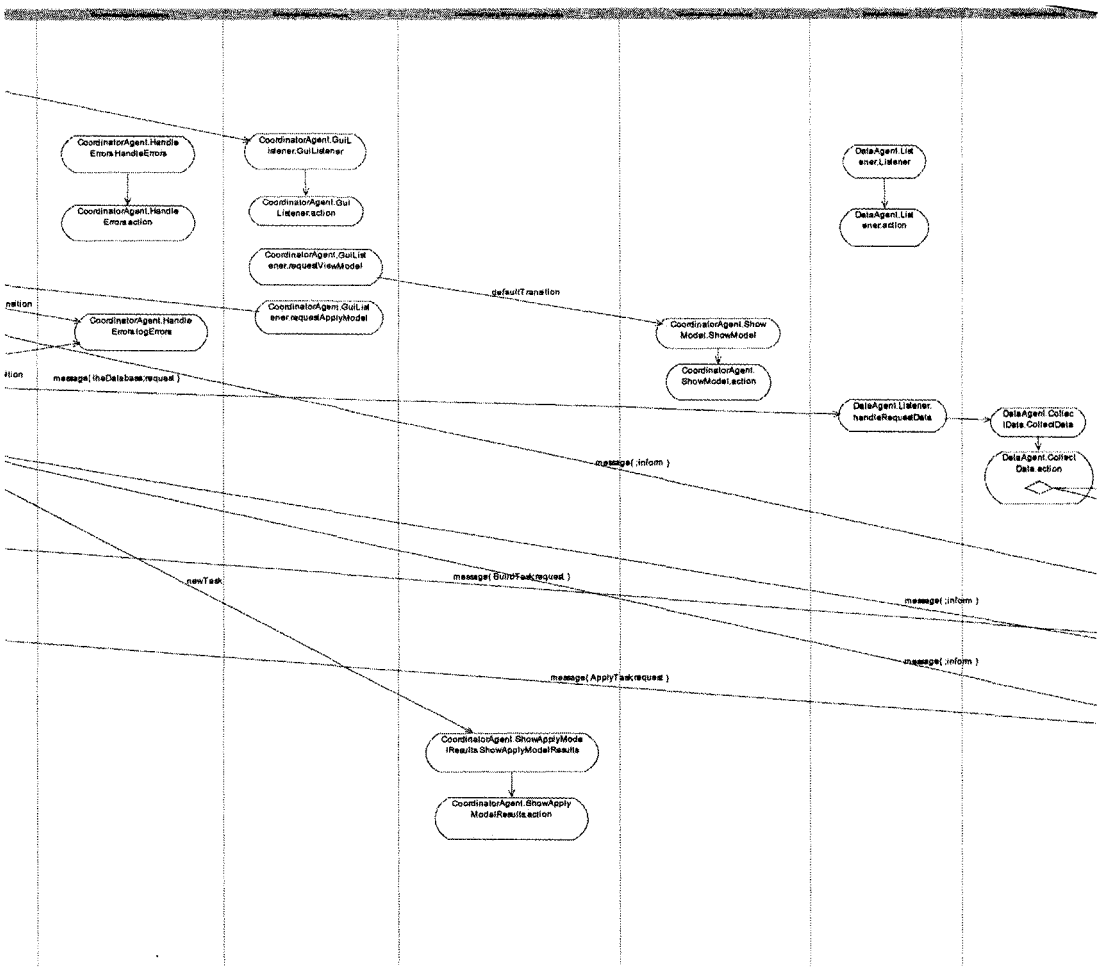


Figure 56 Multi-Agent Behavior Description Diagram (cont.)

As specified by the UML standards, the transitions signify an event (e.g. an incoming message or a task conclusion) or an invocation of methods.

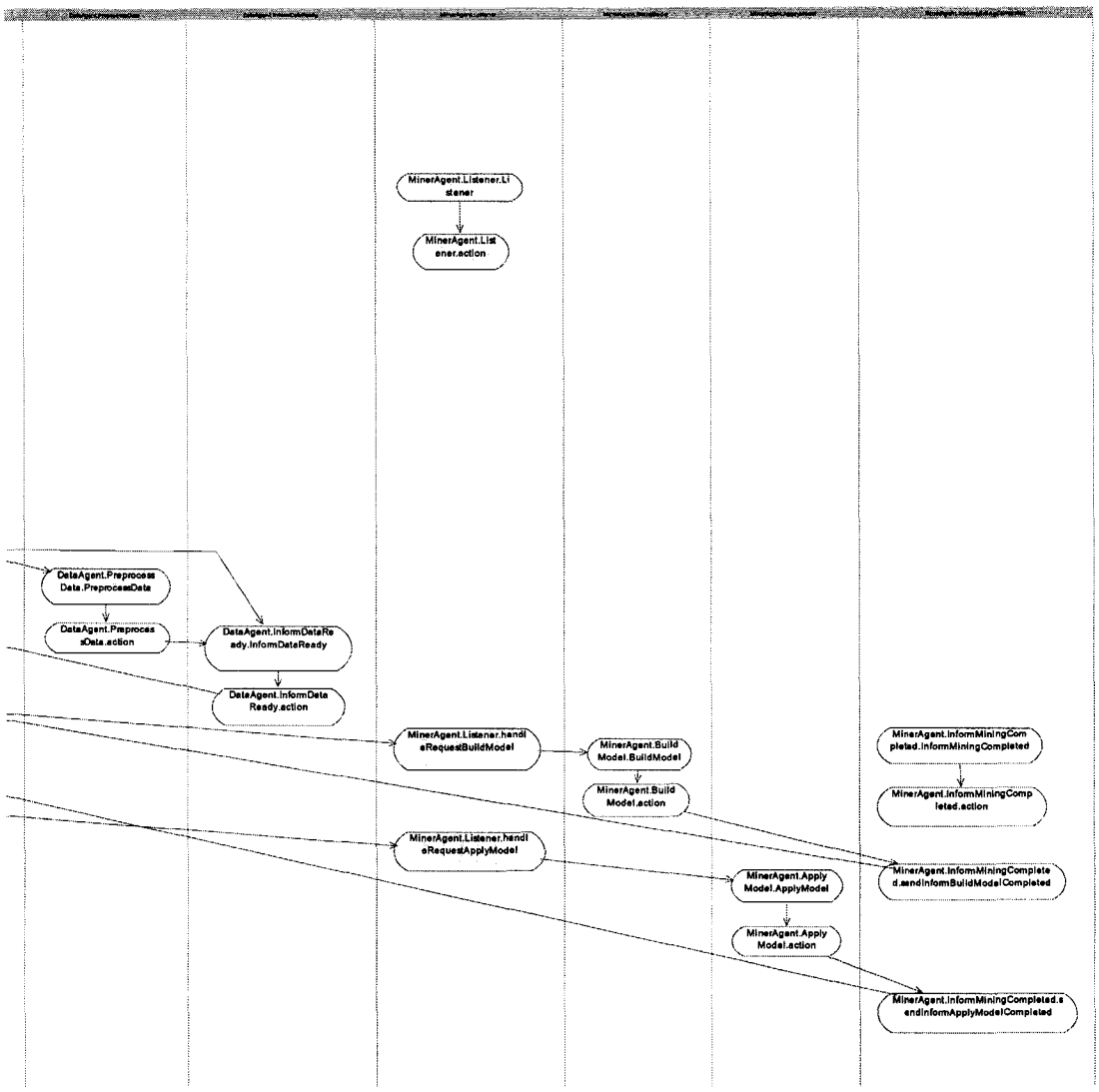


Figure 57 Multi-Agent Behavior Description Diagram (cont.)

The system begin is started by the Coordinator agent. It should first initialize the system with InitializeSystem task. Once the initialization is finished it should start all the cyclic tasks (i.e. all the tasks that will execute forever during lifecycle of the agent), which are MineData, Listener and GuiListener.

The MineData task should first execute collectData() method which should create the RequestData task that will request that data agent prepare a data set for mining. Once

the data agent's Listener task receive the request, it should be processed by the handleRequestData() method and it should create CollectData task. The CollectData task should try to get the raw data from the data source given by the coordinator agent, if the raw data is collected successfully then the PreprocessData task is created and the InformDataReady task is created to inform the coordinator agent that the data collecting and preprocessing is accomplished successfully otherwise the InformDataReady task should inform the failure.

Once the Listener task of the coordinator receives the message of the data agent confirming that data understanding step is finished, it should process the message with the handleDataAgentMsg() method. If an error occurred during the data understanding step the HandleErrors task is created to log the error otherwise the MineData task proceed with the data mining process.

After the end of the data understanding step, the MineData task should proceed with the estimate model step of the data mining process by invoking buildModel() method to create a model. This method should call RequestBuildModel task that should request miner agent to build a model and save it to the MOR. When the Listener task of the miner agent receive the request message, it should process it with the handleRequestBuildModel() method. Afterward, the BuildModel task is invoked to build the model and when the model is build the coordinator agent should be informed with the InformMiningComleted task using sendInformBuildModelCompleted() method.

When the Listener task of the coordinator agent receive the message from the miner agent informing that the build model task is finished, it should process the message with the handleMinerAgentMsg(). If any error occurred during the build model process it should be logged by the HandleErrors task. Otherwise, data agent should be requested to collect another set of data and when the data is collected miner agent should be requested to continue to build the model with new data set and so forth.

User requests are handled by the GuiListener task. The request to apply a model to a set of data that is specified by the user is implemented by the requestApplyModel() method and the request by the user to view a model is implemented by the requestViewModel() method. The requestViewModel() method should execute the ShowModel task to get the model detailed information from MOR and to show the model on GUI. The requestApplyModel() method should inform the MineData task that a apply model task should be executed. Then the applyModel() method should be invoked and it will add RequestApplyModel task to the task execution scheduler. The RequestApplyModel task should send a message to the miner agent to request him to apply a model. The message is received by the Listener task of the miner agent and processed by the handleRequestApplyModel() method. This method in its turn should call ApplyModel task and when the execution of the task is finished it should inform the coordinator agent by sending him a message. The message is send with the sendInformApplyModelCompleted() method of the InformMiningCompleted task. The message is received by the Listener task of the coordinator agent and processed by the handleMinerAgentMsg() method. If a error occurred during the apply model task it should be logged with the logErrors() method of the HandleError task otherwise the apply model results are gotten from MOR and shown on GUI with the ShowApplyModelResults task. Then, the MineData task is informed that the apply model task is completed, so it can use the available miner agent for another build model task.

6.3 Single-Agent Structure Definition

In this section, the structure of each agent is described. The structure is composed of the main agent class and the task classes. The main agent class and the task classes are inherited from the agent class and task class of the selected agent platform, which is in

our case JADE. In JADE, the tasks are represented by the "behaviour" class. The classes defined here correspond to the implementation of the system.

In our system, all tasks are inherited from two types of behaviour jade.core.behaviours.OneShotBehaviour and jade.core.behaviours.CyclicBehaviour classes. Both are extension of jade.core.behaviours.Behaviour class. All tasks that execute only once should be extended from OneShotBehaviour class and the tasks that must be executed forever (e.g. reactive tasks such the Listener task of the agents that wait until a message is received and take action according to the message) should be extended from CyclicBehaviour.

6.3.1 Coordinator Agent

Coordinator agent structure definition is shown in Figure 58.

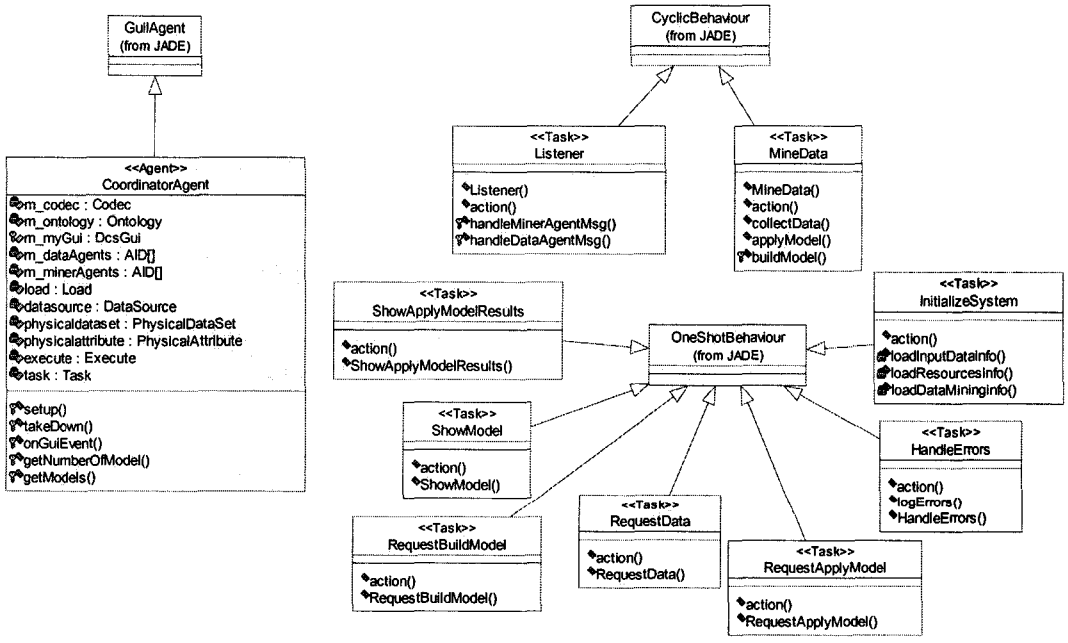


Figure 58 Coordinator Agent Structure Definition

As we mentioned earlier, coordinator agent should interact with the user and JADE platform support agents with a graphical user interface. Unlike the other two agents, the CoordinatorAgent class should inherit from jade.core.GuiAgent, which is a GUI enabled agent class and it is an extension of the jade.core.Agent class.

The coordinator agent should have two tasks that execute forever: Listener and MineData. Listener task should wait for other agent's message and should trigger proper task in reponse to the message. MineData should execute the data mining process incrementally by requesting other agents to do the work.

Also there is one time executed tasks. For example the InitializeSystem should be executed once when the system starts. The request tasks e.g. RequestData, RequestBuildModel and RequestApplyModel are executed when a request is made to other agents. The show tasks e.g. ShowModel, ShowApplyModelResults are executed when the user want to see data mining objects. The HandleErrors should take action when an error occurs.

6.3.2 Data Agent

Data agent structure definition is shown in Figure 59.

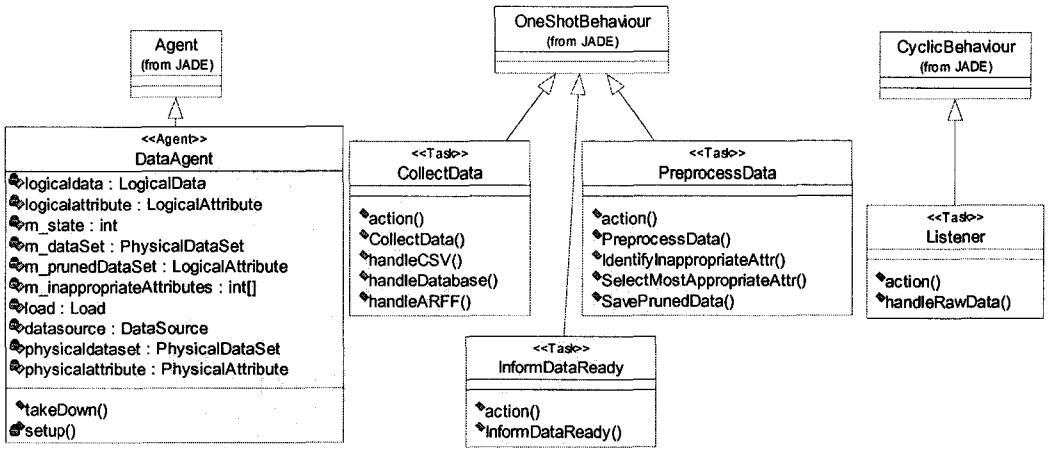


Figure 59 Data Agent Structure Definition

The data agent has four tasks. As for the coordinator agent, the Listener task should wait for messages of the coordinator agents and process them. CollectData should get the data from the specified data source. PreprocessData should perform the data understanding step of the data mining process. InformDataReady should send a message to Coordinator agent that it finished with data loading and preparing.

6.3.3 Miner Agent

Miner agent structure definition is shown in Figure 60.

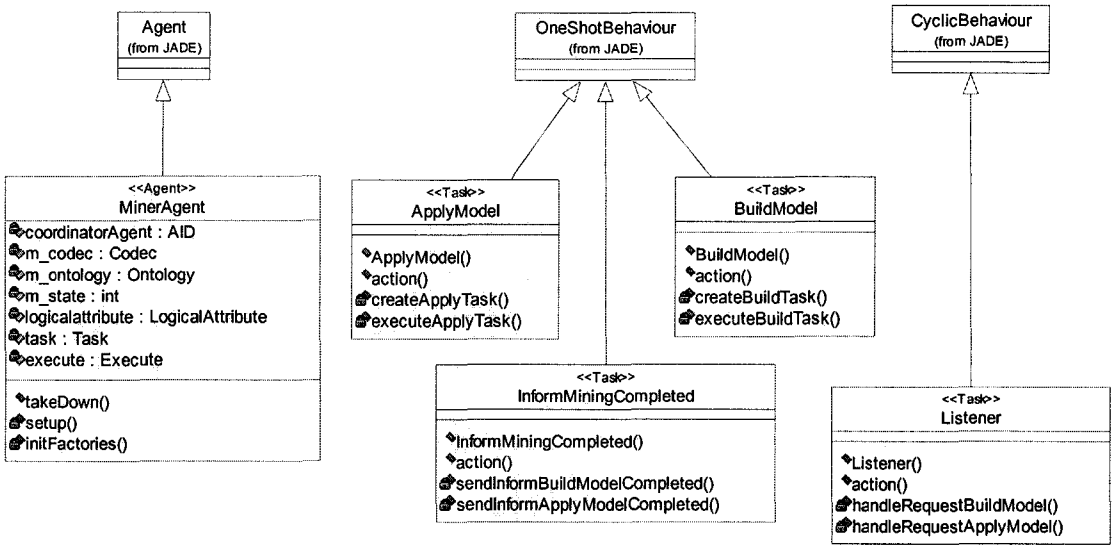


Figure 60 Miner Agent Structure Definition

The miner agent too should have four tasks. Listener task for the incoming messages, BuildModel to build a model, ApplyModel to apply a model on user specified data and InformMiningCompleted to send a message to the coordinator agent that the current mining task (build task or apply task) is accomplished.

6.4 Single-Agent Behavior Description

This section involves the implementation of methods introduced in the previous section (Single-Agent Structure Definition). PASSI don't limit us in describing the algorithm to implement the method. Flow charts, state diagrams as well as semi-formal text descriptions can be used to describe methods implementation.

The implementation details of the agents are in the following subsections.

6.4.1 Coordinator Agent Main Class

A JADE agent is created by extending jade.core.Agent class and implementing the setup() method. In the setup() method, all the agent initializations should be done and the tasks should be scheduled for execution. The initializations are typically setting of the language, the ontology, and the registration with the DF agent (Directory Facilitator is a special agent that provides a Yellow Page service) and other internal variables.

The coordinator agent state diagram is shown in Figure 61.

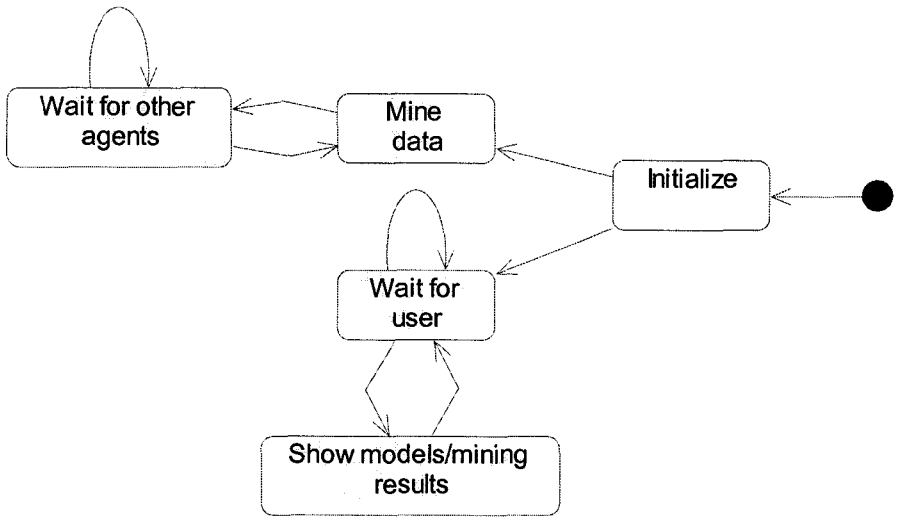


Figure 61 Coordinator agent state diagram

This state diagram doesn't deal with the communication aspect of the coordinator agent with other agents, rather it illustrates only the actions that the coordinator agent will accomplish. All the communication aspects of the agents are described in the section 5.1.2

After the initialization of the agent and the system, the data mining process should begin by scheduling the MineData task. The Listener and GuiListener (e.g. GuiListener is already implemented within GuiAgent, therefore it is started automatically) tasks should also be started. Therefore, the coordinator agent should be waiting for data and miner agents' messages and the GUI interface should be started for user requests. Besides the data mining engine (DME) and mining object repository (MOR), which are used for the data mining several data structures are used to keep track of the mining process. Those data structure are detailed in subsection "6.4.1.1 Tracking data structures".

Basically, the coordinator agents should request (by sending a message) data agent to load and to prepare some raw data and save it into MOR and inform it back. This is accomplished by the RequestData task. When coordinator agent receives the data agent message informing him that the raw data is cleansed and saved into MOR successfully, it should send data agent another message containing information for another cleansing raw data job. The coordinator agent should also request a miner agent to build a model using the cleansed data and save it to MOR. The request for building a model is accomplished by the RequestBuildModel task. When coordinator agent receives a message from the miner agent informing him that he finished its job, the coordinator agent should send him another job and so forth. All messages send by other agents (data and miner agents) are handled by the Listener task

The user interacts with the coordinator agent through the GUI using jade.gui.GuiEvent. When the GuiListener receives an APPLY_EVENT event, it should update the apply tracking data structure and schedule the MineData task to apply the model on the raw data that user specified. The apply model is realized by miner agent; therefore the coordinator agent sends a message to the miner using RequestApplyModel task for that purpose. Once the miner agent is finished with the apply model job, it sends back a message to the coordinator agent then the coordinator agent shows the results of the apply model on the GUI with the ShowApplyModelResults task. If coordinator agent

receives a SHOW_MODEL_EVENT event, it simply shows the model using ShowModel task.

Whenever an error occurs while an agent realize its requested job, it should inform it to the coordinator agent and the coordinator agent should log it with the HandleErrors task. Further details on agent's tasks are given in section 6.4.2.

The coordinator agent should be in only one of the states in Table VII.

Table VII

Coordinator agent's states

State	Description
DM_ACTIVE	The DCS system is active.
DM_STOPPED	The user requested to stop DCS therefore the data mining process should stop and system shutdown process should begin.

6.4.1.1 Tracking data structures

The coordinator agent should use several data structure to carry out the data mining process. There should be four data structures.

- Build task tracking data structure
- Apply task tracking data structure
- Data agents tracking table
- Miner agent tracking table

Build and apply tracking data structures should be saved in text files and when the system is restarted, it should update its data structures according to those files.

Table VIII

Build task tracking data structure

Data Source					Pruned Data		Build Model					
DB name	DB URL	DB type	1st index (optional)	last index (optional)	pruned (yes/no)	pruned data name	PART					•••
							build (yes/no)	model name	suite (yes/no)	source model name	Setting name	•••

Build task tracking data structure is composed of three parts: data source, pruned data and build model. Data source section hold information related to data source such as the name, the URL, the type, etc. Pruned data section is used to know if the data is loaded and cleansed by the data agent. Pruned option informs us that the raw data is ready for mining and pruned data name is the name of the pruned data in MOR. Build model section is used to keep track of the model building. Therefore, in build model section for each type of model that DCS system can build there should be five elements. For example, in Table VIII for PART algorithm, build option is used to know if the model is build or not build, model name is the name of the PART model saved in MOR, suite option informs us if the model that is build is continuation of another model or it is a new build (e.g. we can build a model with one data and continue building the same model with another data or build a new model for every data). If the current model is the continuation of another model, the source model name is the name of the model in MOR from which the current model is build. Setting name is name of model setting object in MOR.

Table IX

Apply task tracking data structure

Data Source					Pruned Data		Apply Model				
DB name	DB URL	DB type	1st index (optional)	last index (optional)	pruned (yes/no)	pruned data name	apply (yes/no)	Source model name	Setting name	Setting parameters	Output name

The apply task has priority on build task. The apply task tracking data structures is updated every time the user request an apply task from the GUI. It is composed of three sections where the first two sections are similar to the Build task tracking data structure. In the apply model section, the apply option informs us if the apply is done. Source model name is the input model. The setting name is the name of the apply setting and the setting parameters subsection contains all apply setting parameters. The output name is the name of the output data.

Table X

Model setting data structure

Model name	**Nb. parameters**	**Parameters**
PART	15	P1;P2;...

The structure in Table X is used to store the model setting parameters. Every time, miner agent builds a model it should use the setting information stored in this structure.

Table XI

Data agents tracking table

Agent ID	Available (yes/no)

Table XII

Miner agent tracking table

Agent ID	Available (yes/no)

Table XI and Table XII are used to know which agent is available for a data mining job. Those tables are initialized by the InitializeSystem task and updated every time coordinator agent requests a job to a data or miner agent and those agents finish their job.

6.4.1.2 GUI Interface

When implementing a GUI, which has its own thread to handle events generated by the user with a multi agent system where each agent has their own execution thread can be difficult to do. Therefore, JADE provides a mechanism to manage the user interactions with the multi agent system, for that reason JADE includes GUI integrated agent class jade.gui.GuiAgent. This mechanism is based on event passing (i.e. GUI and agent communicate with each other by passing events).

The GUI is implemented by extending javax.Swing.JFrame class. In this document, we won't get into detail how a GUI works or implemented in Java programming language instead the interaction between agent and GUI will be described.

A GUI has a built-in mechanism to handle event generated every time a user interact with GUI by pressing a button or entering a value. This mechanism is implemented by the actionPerformed() method of an ActionListener object. Every time a component of the GUI registered with the ActionListener object is invoked, an ActionEvent is generated and the actionPerformed() method is called to handle the ActionEvent. The agent also can interact with GUI using this mechanism.

Therefore, GUI interface should implement ActionListener object and each component of DCS's GUI (DcsGui) should be registered using addActionListener() method. The implementation details of the actionPerformed() method are given in section 6.4.1.2.1.

The mechanism described above allows a user or an agent to interact with the GUI but what we also need is the inverse: the GUI to interact with the agent. To do that JADE provided the abstract class GuiAgent. This class has two methods that allow a GUI to communicate to an agent program: postGuiEvent() and onGuiEvent(). The postGuiEvent() method is used to send an event (GuiEvent object) by GUI to the agent and onGuiEvent() method handles the events received by the agent.

Therefore, the coordinator agent which has GUI should extend the GuiAgent class. In the GUI implementation, GuiEvent object is created and posted to the agent with the postGuiEvent() method. The onGuiEvent() method should be implemented according to the section 6.4.1.2.2

Further details on GUI enabled agent implementation with JADE platform can be found in [47].

The possible events in our implementation are described in Table XIII:

Table XIII

GUI Events

Event	Description
QUIT_EVENT	This event is triggered to end the DCS.
REFRESH_EVENT	This event is triggered to refresh the list of model produced by DCS shown on GUI.
SHOW_MODEL_EVENT	This event is triggered to show a model selected from the list of model shown on GUI.
APPLY_EVENT	This event is triggered to apply a model selected from the list of model shown on GUI on a data set identified on the user interface.

6.4.1.2.1 actionPerformed()

This method is member of DcsGui object and should have an ActionEvent object as parameter. It is implemented as several overlapping if-else condition (or a switch) where each "if" condition verify the event type and process the corresponding event. For each ActionEvent, and GuiEvent is send to the agent. A GuiEvent object has two mandatory attributes and an optional list of parameters. The mandatory attributes are the source of the event and the type of the event.

When the actionPerformed() method is invoked by QUIT_EVENT or REFRESH_EVENT ActionEvent (i.e. generated when the user click on "quit" button or on "refresh" button) their corresponding GuiEvents are created and posted with the postGuiEvent().

With the SHOW_MODEL_EVENT ActionEvent, the actionPerformed() method should create a SHOW_MODEL_EVENT GuiEvent and add the name of the model selected by the user as optional parameter to the GuiEvent before posting it.

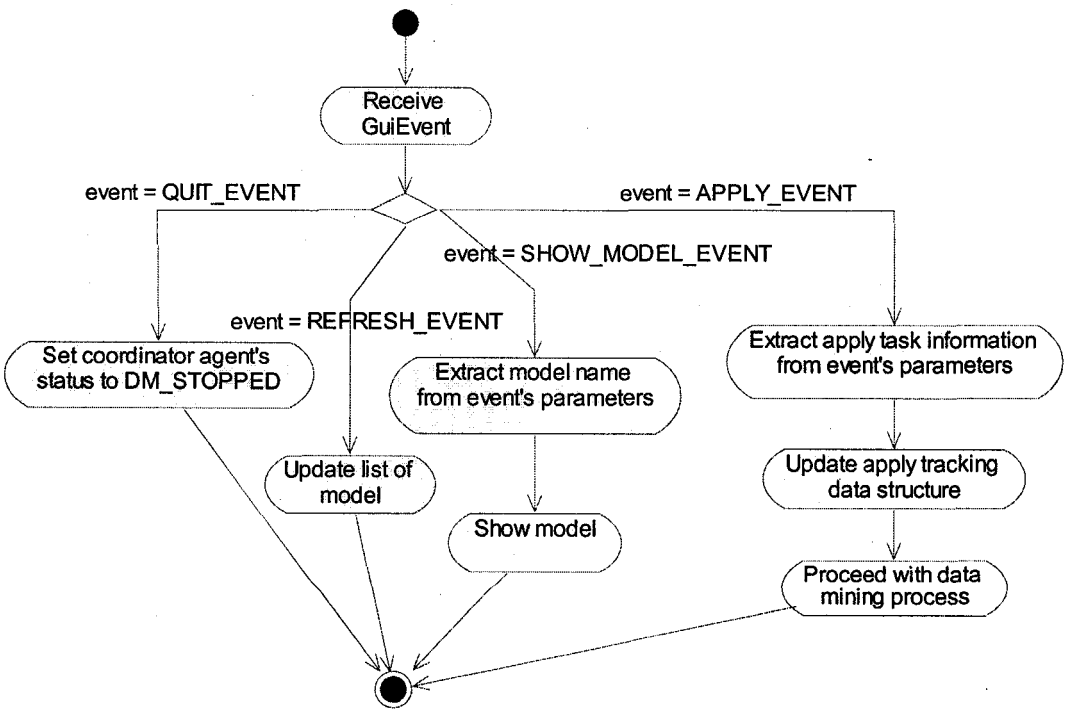
With the APPLY_EVENT ActionEvent, the actionPerformed() method should create an APPLY_EVENT GuiEvent and add the following optional parameters:

- input model,
- input data
- apply setting parameters (i.e. output data specifications) and
- output data name.

Then the GuiEvent is posted.

6.4.1.2.2 onGuiEvent()

This method is member of the coordinator agent and it process the GuiEvent received by this agent. The algorithm of onGuiEvent() method is shown in Algorithm 8.



Algorithm 8 Coordinator::onGuiEvent() method's algorithm

If the event is QUIT_EVENT then the coordinator agent state is changed to DM_STOPPED. If the event is REFRESH_EVENT then the list of model shown on GUI is refreshed by the RefreshModelList task. If the event is SHOW_MODEL_EVENT then the name of the selected model is retrieved from the message and it is shown on GUI using ShowModel task. If the event is APPLY_EVENT then all optional parameters are extracted and the apply task tracking data structure is updated and we proceed with the data mining process by calling the MineData task.

6.4.2 Coordinator Agent Tasks

The coordinator agent has the following ten tasks:

- InitializeSystem task
- Listener task
- MineData task
- RequestData task
- RequestBuildModel task
- RequestApplyModel task
- HandleErrors task
- ShowModel task
- ShowApplyModelResults task

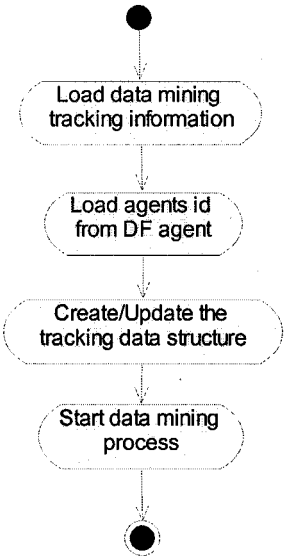
In JADE, each task implementation is an extension of Behaviour class and it must implement the action() method. This method defines the operations that the task performs when it is scheduled for execution.

6.4.2.1 InitializeSystem task

This task should create and initialize the tracking data structures of the coordinator agent. All the information that those structures contains are loaded from text files. This task consists of one method:. action().

6.4.2.1.1 action()

The algorithm of InitializeSystem task's action() method is shown in Algorithm 9.



Algorithm 9 Coordinator::InitializeSystem::action() method's algorithm

First build task tracking data structure is created and initialized by loading the information from a text file. Afterward the data agent tracking table and miner agent tracking table are created and updated, by getting the AID (JADE agent identifier) of all miner and data agents available on the platform, registered with the DF. Then the MineData task is scheduled for execution.

6.4.2.2 Listener task

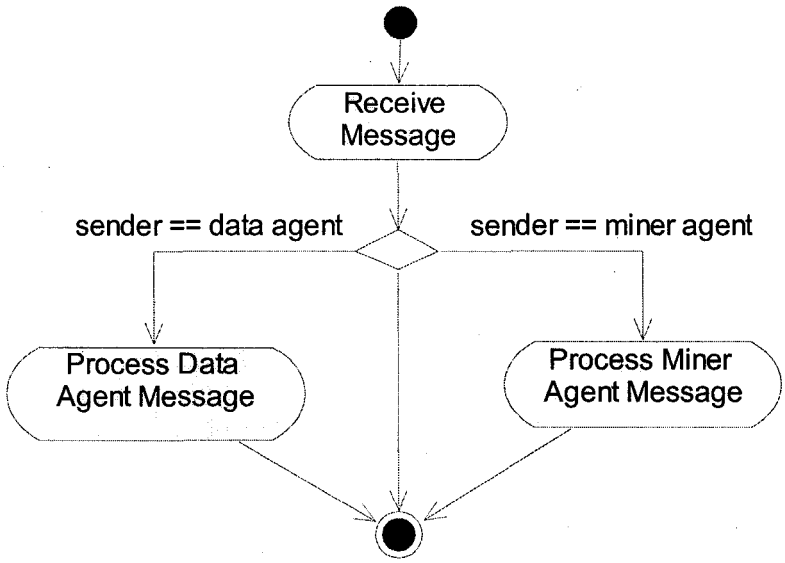
This task receives the messages from data and miner agents and processes it. The received message is "inform" communicative act and its content is described in section 5.1.2.

This task consists of three methods:

- action()
- handleDataAgentMsg()
- handleMinerAgentMsg()

6.4.2.2.1 action()

The algorithm of Listener task's action() method is shown in Algorithm 10.



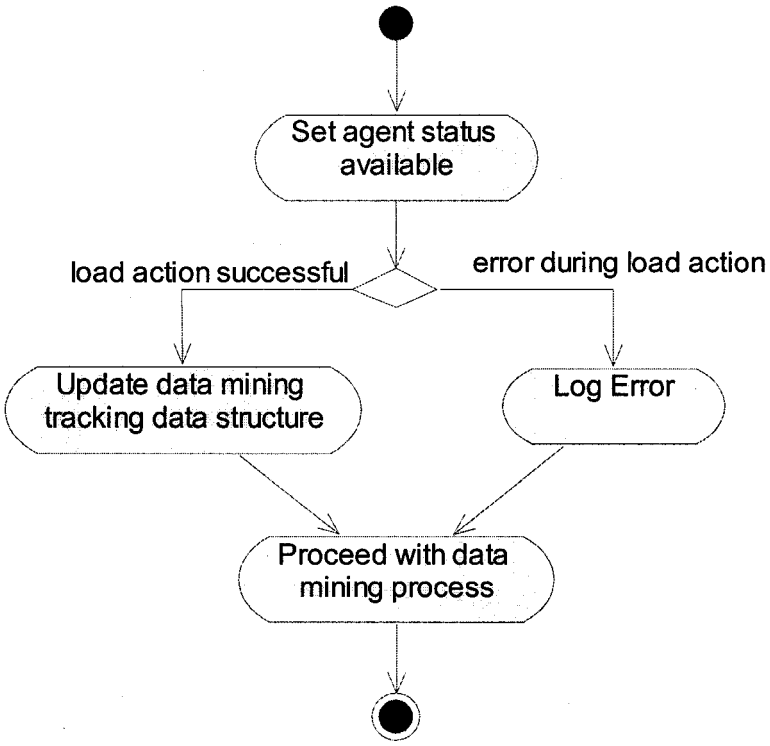
Algorithm 10 Coordinator::Listener::action() method's algorithm

This method should be implemented as an infinite loop (as well as the Listener's task of other agents). First, the received message's content is extracted. If the sender is a data agent then the message is processed by the "process data agent message" activity and if the sender is miner agent then the message is processed by the "process miner agent message" activity. The "process data agent message" activity is achieved by the handleDataAgentMsg() method and the "process miner agent message" activity is

achieved by the handleMinerAgentMsg () method both described in the following sections 6.4.2.2.2 and 6.4.2.2.3.

6.4.2.2.2 handleDataAgentMsg()

The algorithm of Listener task's handleDataAgentMsg() method is shown in Algorithm 11.



Algorithm 11 Coordinator::Listener::handleDataAgentMsg method's algorithm

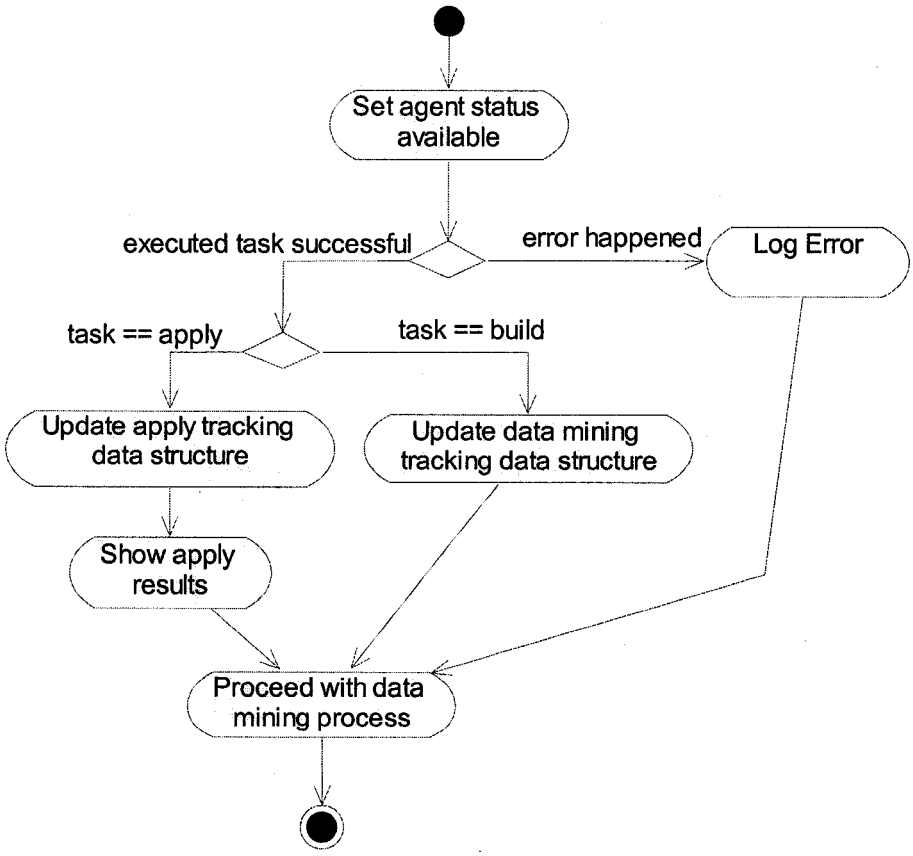
First the data agent status is set to available by setting the available option to "yes" in Table XI

Data agents tracking table. If the load action is completed properly then the pruned

option in pruned data section of the corresponding tracking data structure (i.e. if the data is for a build task then the build task tracking data structure is updated) is set to "yes". If the load action isn't accomplished properly then HandleError task is scheduled to log the error. Then, we proceed with the data mining process using the MineData task.

6.4.2.2.3 handleMinerAgentMsg()

The algorithm of Listener task's handleMinerAgentMsg() method is shown in Algorithm 12.



Algorithm 12 Coordinator::Listener::handleMinerAgentMsg() method's algorithm

First the miner agent status is set to available by setting the available option to "yes" in Table XII

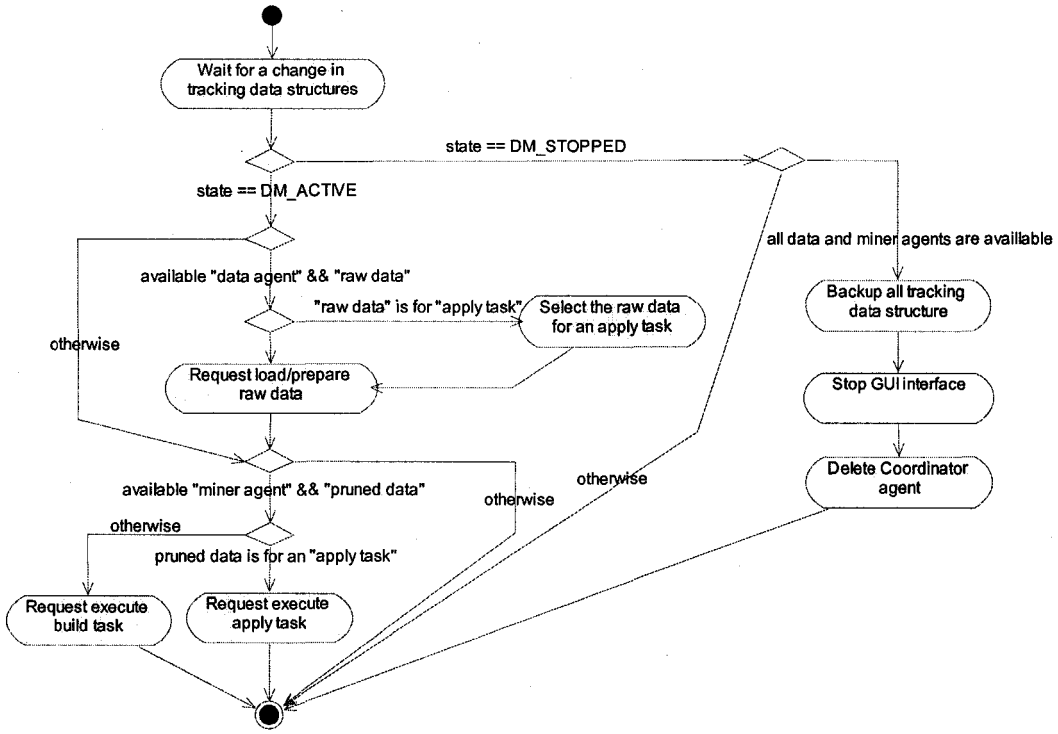
Miner agent tracking table. If the execute action is completed properly and the executed task is an apply task then the apply option in apply model section of the apply task tracking data structure is set to "yes" and ShowApplyModelResults task is scheduled to show the apply results. If the build task is executed successfully then the build option in build model section of the build task tracking data structure is set to "yes". If an error occurred during the execute task then HandleError task is scheduled to log the error. Then, we proceed with the data mining process using the MineData task.

6.4.2.3 MineData task

The data mining process is realized by this task. This task should keep track of all data mining activities realized by other agents (data/miner agents). This task consists of one main method: action().

6.4.2.3.1 action()

The algorithm of MineData task's action() method is shown in Algorithm 13.



Algorithm 13 Coordinator::MineData:action() method's algorithm

This method should be implemented as an infinite loop called (unblocked) every time there is a change in the tracking data structures described in section 6.4.1.1Tracking data structures. If the state of the coordinator agent is DM_STOPPED then it should initiate the system stopping activities which are backing up all the tracking data structure by saving them into text files, stopping the GUI and deleting the coordinator agent. Otherwise, it should proceed with the data mining activities.

The apply task has priority over build task to improve the user responsiveness. Therefore, if there is available data agent and raw data to be cleansed, the raw data for an apply task should be selected first. Once the raw data that will be prepared by the data agent is known, the data agent should be requested to load and prepare the raw data.

This action is accomplished by the RequestData task. This task should also update the Data agents tracking table and the corresponding mining task tracking data structure. If there is an available miner agent and pruned data then a mining task should be requested using RequestApplyModel or RequestBuildModel. Once again, if there are several pruned data available, the pruned data related to an apply task should have priority over pruned data for a build task.

With the build task, if a model to be built is continuation of another model, then before requesting a build model, the source model must be ready even if the cleansed data is available.

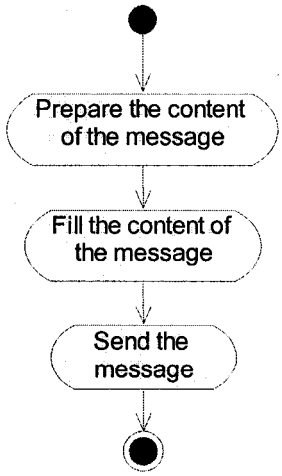
When there is more than one available data or miner agent then the first available agent from the agent tracking table should be selected.

6.4.2.4 RequestData task

This task defines all the operations to send a request message to a data agent to load and prepare a set of raw data for mining later. This task is scheduled from the MineData task. This task consists of one method: action().

6.4.2.4.1 action()

The algorithm of RequestData task's action() method is shown in Algorithm 14.



Algorithm 14 Coordinator::RequestData:action() method's algorithm

The message is prepared by setting the following parameters of the message:

- sender: the AID (agent identifier) of the coordinator agent
- receiver: the AID of the data agent
- language: RDF is used in our implementation
- ontology: DCSOntology as described in section 5.1.1.

DCSOntology is implemented by extending the jade.content.onto.Ontology according to Domain Ontology Description Diagram. In [48], it gives some information on how to use and ontology.

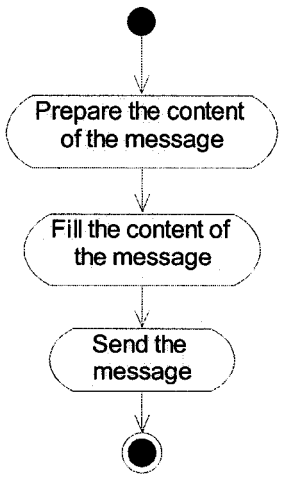
Then, the content of the message is filled with the "load" action as defined in section 5.1.1.

6.4.2.5 RequestBuildModel task

This task defines all the operations to send a request message to a miner agent to build a model from a pruned data set. This task is scheduled from the MineData task. This task consists of one method: action().

6.4.2.5.1 action()

The algorithm of RequestBuildModel task's action() method is shown in Algorithm 15.



Algorithm 15 Coordinator::RequestBuildModel:action() method's algorithm

The message is prepared by setting the following parameters of the message:

- sender: the AID (agent identifier) of the coordinator agent
- receiver: the AID of the miner agent
- language: RDF is used in our implementation
- ontology: DCSOntology as described in section 5.1.1.

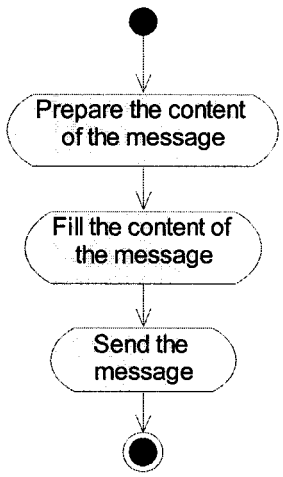
Then, the content of the message is filled with the "execution" action with a "build task" as defined in section 5.1.1.

6.4.2.6 RequestApplyModel task

This task defines all the operations to send a request message to a miner agent to apply a model on a pruned data set. This task is scheduled from the MineData task. This task consists of one method: action().

6.4.2.6.1 action()

The algorithm of RequestApplyModel task's action() method is shown in Algorithm 16.



Algorithm 16 Coordinator::RequestApplyModel::action() method's algorithm

The message is prepared by setting the following parameters of the message:

- sender: the AID (agent identifier) of the coordinator agent
- receiver: the AID of the miner agent
- language: RDF is used in our implementation

- ontology: DCSOntology as described in section 5.1.1.

Then, the content of the message is filled with the "execution" action with a "apply task" as defined in section 5.1.1.

6.4.2.7 HandleErrors task

This task should handle the errors that happen during the data mining process. For now, the only action taken by this task is to log errors into a text file. The format of the logging is shown in Figure 62.

Data/time (dd-MM-YYYY / ss:mm:hh)	Agent AID	Agent type (data / miner)	Error

Figure 62 Error logging format

This task is scheduled for execution from the Listener task and it consists of one method: action().

6.4.2.8 ShowModel task

This task shows the model that is selected by the user on GUI. First, the selected model is retrieved from MOR and passed to the GUI. Then, it is shown on the GUI. This task is triggered from the GUI and it consists of one method: action().

6.4.2.9 ShowApplyModelResults task

This task shows the apply task results that was requested by the user from GUI. When the miner agent that executes the apply task has finished, it should send a message the

coordinator agent confirming the end of it apply task job. Then, the apply results are retrieved from MOR by the coordinator agent and passed to the GUI. After that, they are shown on the GUI. This task is triggered from the coordinator agent's Listener task and it consists of one method: action().

6.4.2.10 RefreshModelList task

This task should refresh the list of models shown on GUI. First, the name of all available models are retrieved from MOR and passed to the GUI. Then, they are shown on the GUI. This task is triggered from the GUI and it consists of one method: action().

6.4.3 Data Agent Main Class

The data agent state diagram is shown in Figure 63.

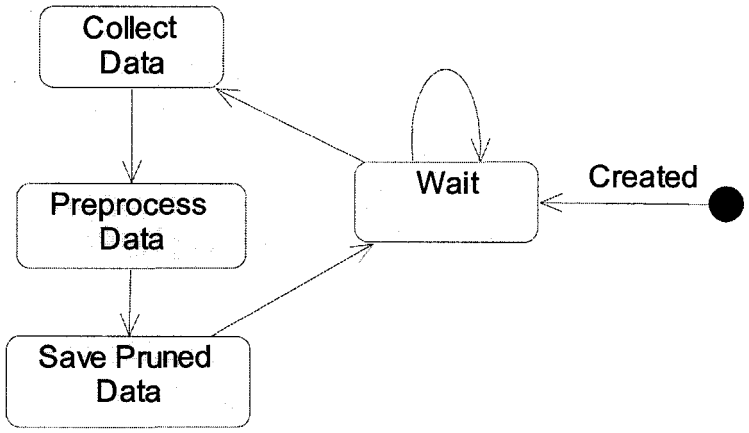


Figure 63 Data agent state diagram

After the initialization of the agent, the Listener task should be scheduled and the agent should be waiting for coordinator agent's message. When the message is received and

extracted, the raw data should be collected with the CollectData task from the data source and cleansed with the PreprocessData task. Then, the cleansed data should be saved in MOR and the data agent should pass to waiting mode after informing the coordinator agent the end of its job. Further details on agent's tasks are given in the section 6.4.4.

The data agent should be in only one of the states in Table XIV.

Table XIV

Data agent states

State	Description
DATA_SAVED	The load and prepare actions are finished and data is saved
ERROR_COLLECTDATA	An error occurred during the data collecting task.
ERROR_PREPROCESSDATA	An error occurred during the data preprocessing task
ERROR_SAVEPRUNEDDATA	An error occurred during the data saving
NOT_FINISHED	The load and prepare actions are not finished

6.4.4 Data Agent Tasks

The data agent has the following four tasks:

- Listener task
- CollectData task
- PreprocessData task
- InformDataReady task

6.4.4.1 Listener task

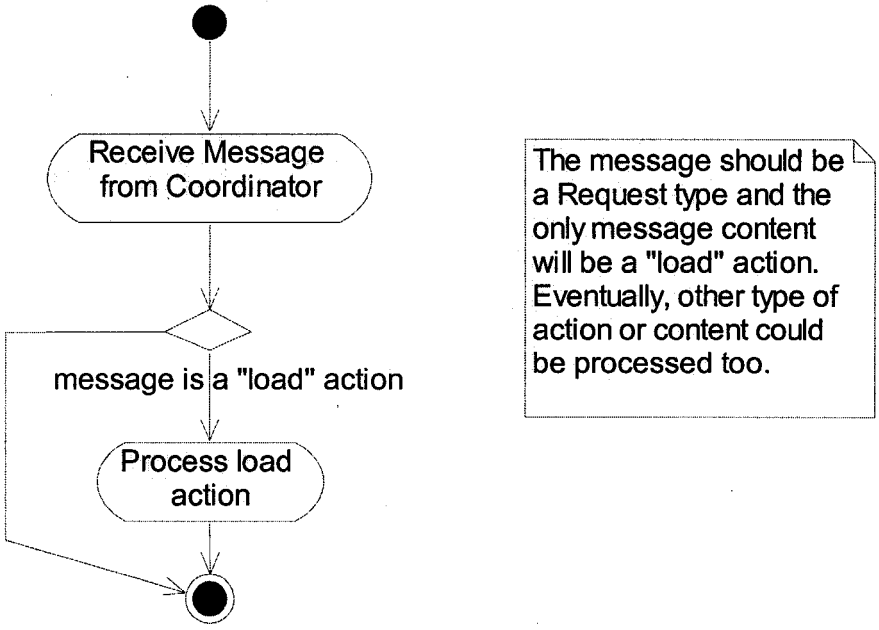
This task receives the messages from coordinator agent and processes it. The received message is "request" communicative act and its content is a "load" action taken from domain ontology.

This task consists of two methods:

- action()
- handleRawData()

6.4.4.1.1 action()

The algorithm of Listener task's action() method is shown in Algorithm 17.

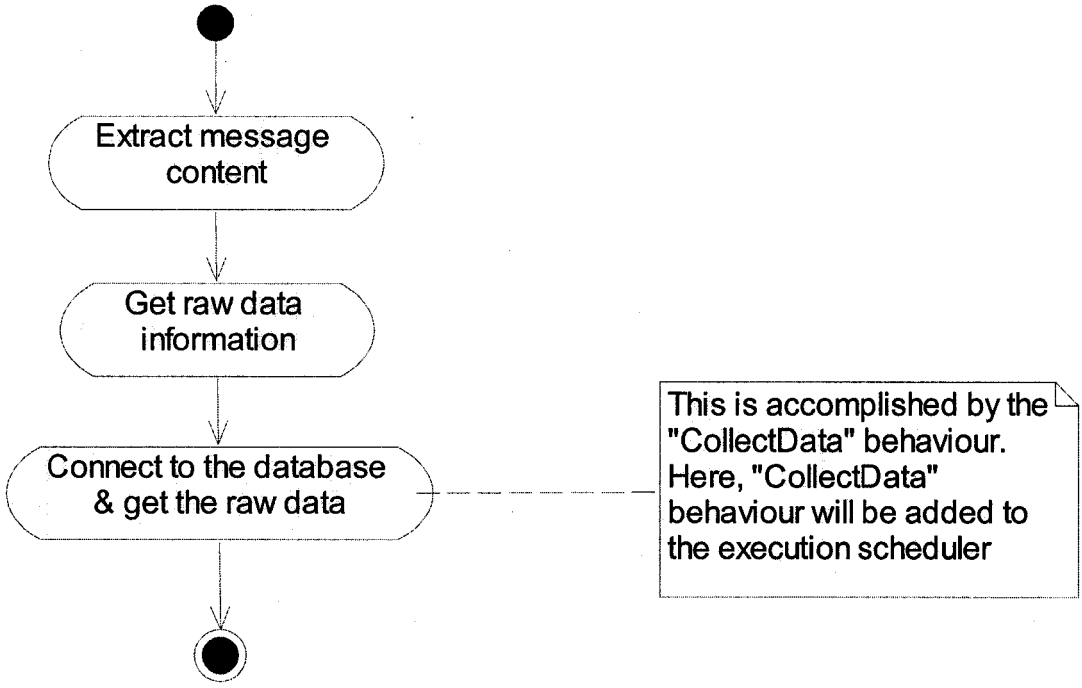


Algorithm 17 Data::Listener:action() method's algorithm

First, the message received from the coordinator agent content is extracted. If the content is a load action, then "process load action" activity is realized by the handleRawData() method described in section 6.4.4.1.2.

6.4.4.1.2 handleRawData()

The algorithm of Listener task's hadleRawData() method is shown in Algorithm 18.



Algorithm 18 Data::Listener::handleRawData() method's algorithm

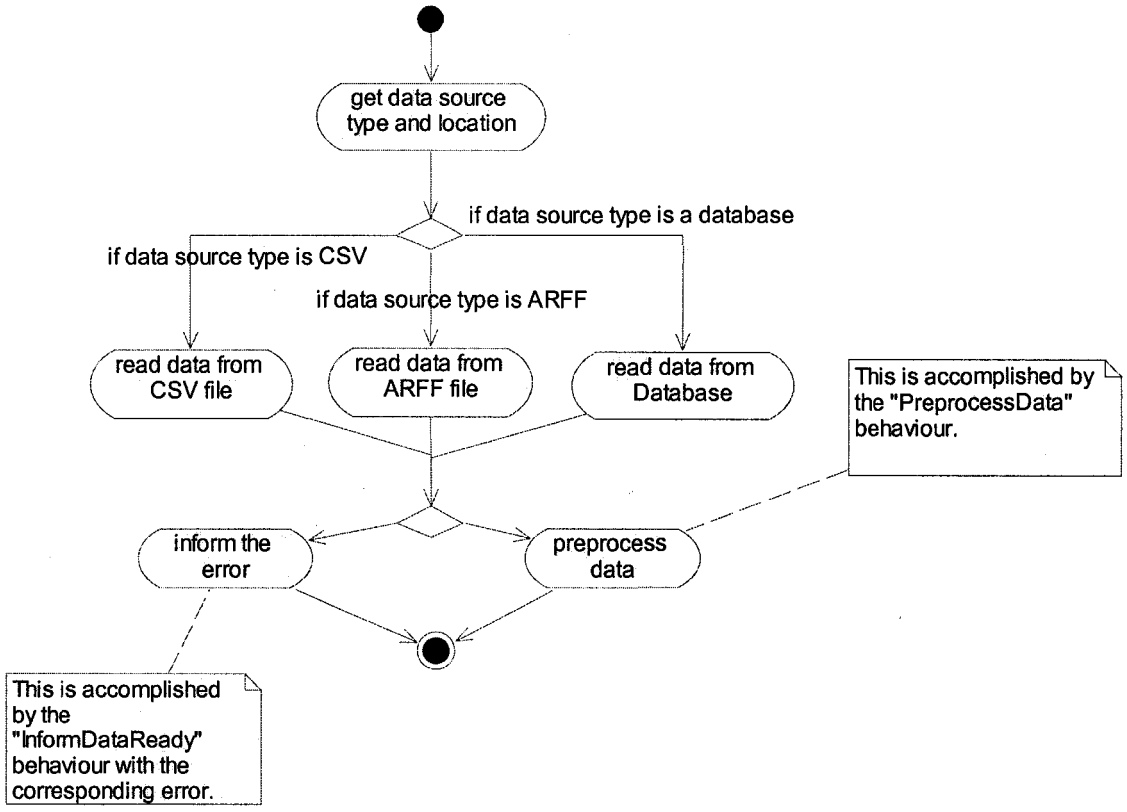
This method should extract the content of the received message and get the information on the raw data. The information is the location of the raw data, the format (i.e. csv file or database or any other format), etc. and it is represented by the "DataSource" concept as shown in the domain ontology description. Then CollectData task is scheduled for execution to collect the data.

6.4.4.2 CollectData task

This task should gather the raw data using the data source information. This task consists of one method: action().

6.4.4.2.1 action()

The algorithm of CollectData task's action() method is shown in Algorithm 19.



Algorithm 19 Data::CollectData::action() method's algorithm

The format and the location of the raw data are obtained from the data source. If the raw data is read properly then the data should be cleansed using the PreprocessData task otherwise it should finish it processing and pass in wait mode after sending a message to coordinator agent using InformDataReady task. The message should be an "inform"

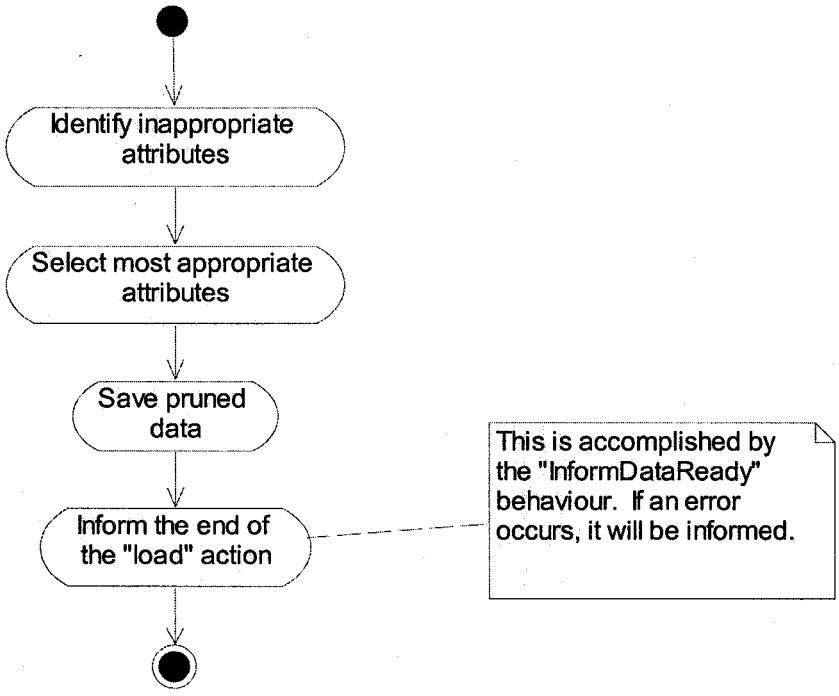
communicative act and it should contain the type of error that occurred as specified in section 5.1.2.1.

6.4.4.3 PreprocessData task

This task cleanses the data according to the section 4.1.2Data understanding. This task consists of one method: action().

6.4.4.3.1 action()

The algorithm of PreprocessData task's action() method is shown in Algorithm 20.



Algorithm 20 Data::PreprocessData::action() method's algorithm

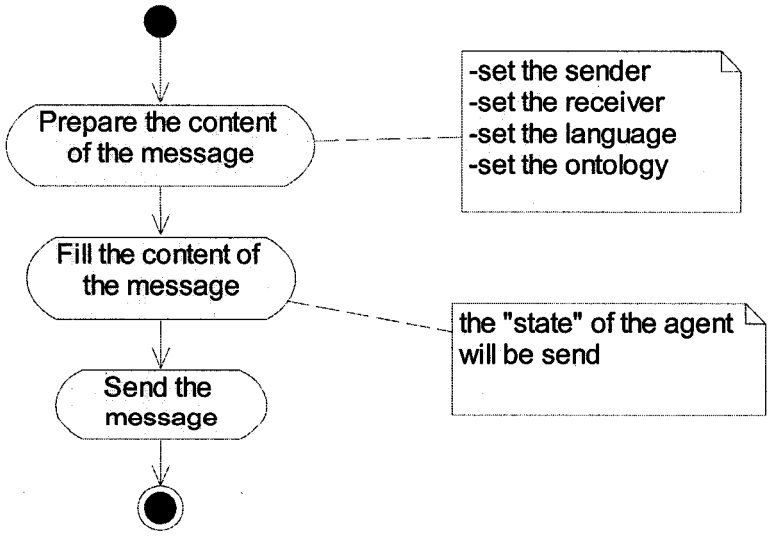
"Identify inappropriate attributes" and "Select most appropriate attributes" activities are described in detail in sections 4.1.2.1and 4.1.2.2respectively. The pruned data should be saved in MOR. The name of the pruned data is given by the coordinator agent and it is contained in the initial message. The pruned data should be a PysicalDataSet concept (see domain ontology description diagram). The end of the processing should be informed to coordinator agent using InformDataReady task.

6.4.4.4 InformDataReady task

InformDataReady task should send a message to coordinator agent informing the end of data agent processing. This task consists of one method: action().

6.4.4.4.1 action()

The algorithm of Listener task's action() method is shown in Algorithm 21.



Algorithm 21 Data::InformDataReady::action() method's algorithm

The message is prepared by setting the following parameters of the message:

- sender: the AID (agent identifier) of the data agent
- receiver: the AID of the coordinator agent
- language: RDF is used in our implementation
- ontology: DCSOntology as described in section 5.1.1.

Then, the content of the message is filled with the "state" of the agent and the message is send.

6.4.5 Miner Agent Main Class

The miner agent state diagram is shown in Figure 64.

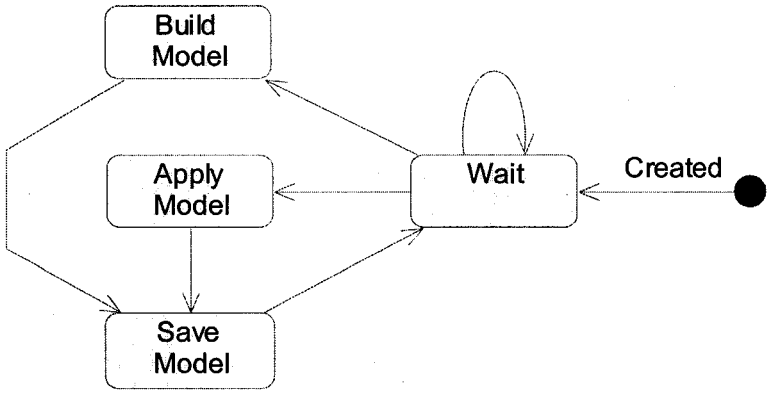


Figure 64 Miner agent state diagram

The miner agent should accomplish two tasks: build model and apply model. Then, the results of the mining task should be saved in MOR and the agent should wait for the next request of the coordinator agent. Further details on agent's tasks are given in section 6.4.6.

The miner agent should be in only one of the states in Table XV.

Table XV

Miner agent states

State	Description
MINING_DONE	The apply or build task is successfully completed.
ERROR_BUILDMODEL	An error occurred during the build model task.
ERROR_APPLYMODEL	An error occurred during the apply model task
NOT_FINISHED	The apply or build task is not completed.

6.4.6 Miner Agent Tasks

The miner agent has the following four tasks:

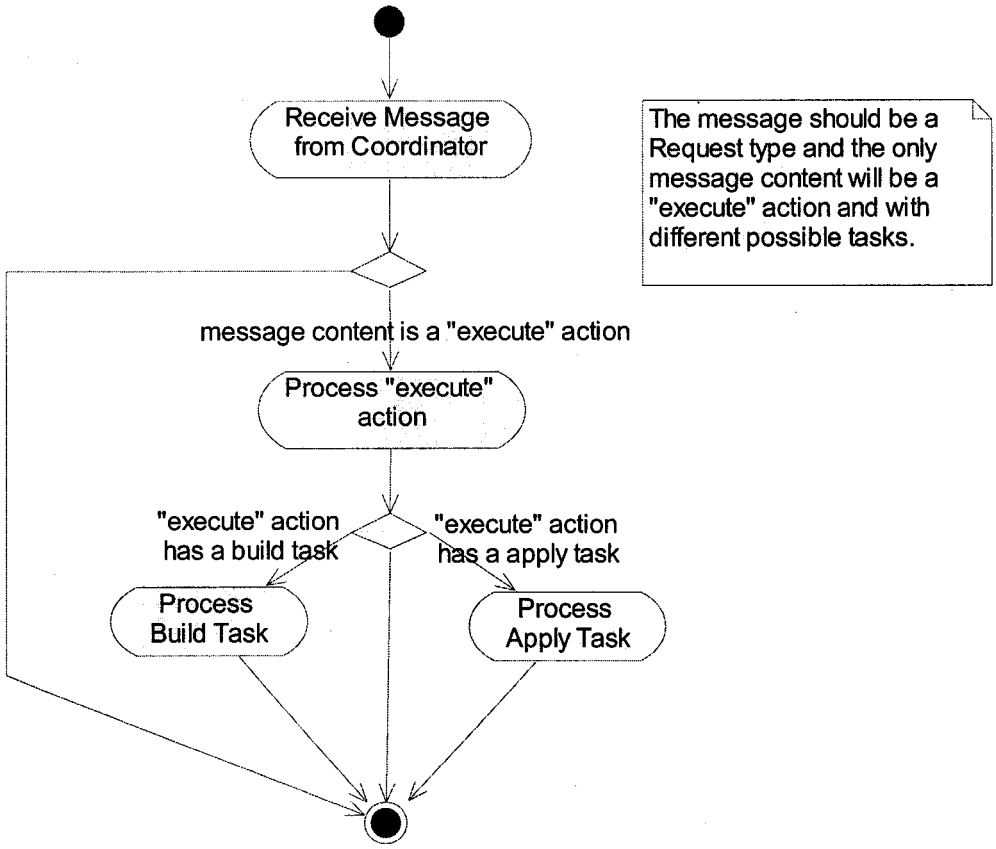
- Listener task
- BuildModel task
- ApplyModel task
- InformMiningCompleted task

6.4.6.1 Listener task

This task receives messages from coordinator agent and processes it. The received message is "request" communicative act and its content is an "execution" action taken from domain ontology. This task consists of one method: action().

6.4.6.1.1 action()

The algorithm of Listener task's action() method is shown in Algorithm 22.



Algorithm 22 Miner::Listener::action() method's algorithm

First, the message received from the coordinator agent content is extracted. If the content is a "execute" action, then the content of the "execute" action object is processed. If the "execute" action has a build task then "Process Build Task" is realized by the BuildModel task, otherwise if the "execute" action has an apply task then "Process Apply Task" is realized by the ApplyModel task.

6.4.6.2 BuildModel task

This task builds a model following the algorithm "Model building scenario" in section "5.4 Step 4 – Estimate the model". Further details on how to build a model using JSR 73: JDM compliant DME can be found in document [14].

6.4.6.3 ApplyModel task

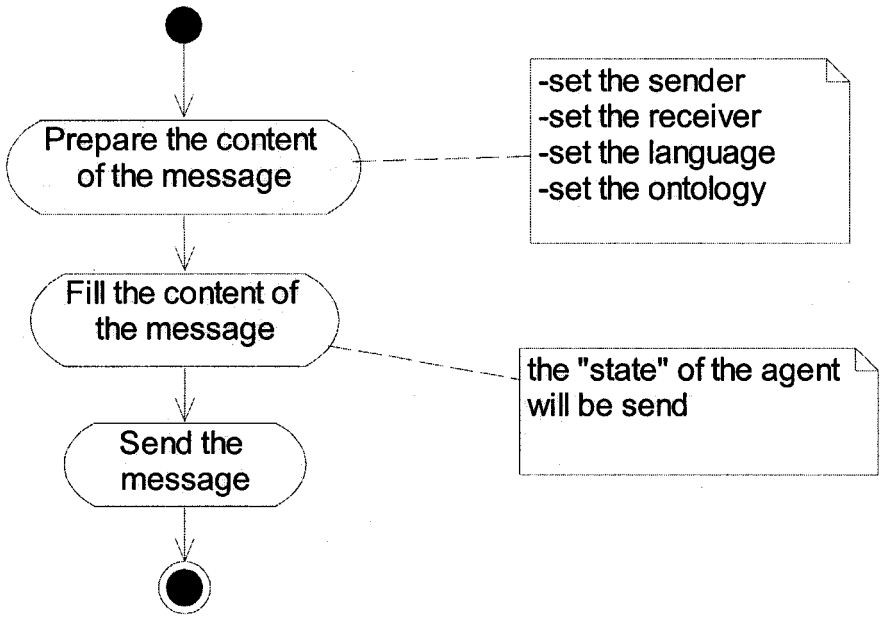
This task applies a model on data both specified from GUI by the user following the "Algorithm 7 Apply building scenario" in section "5.4 Step 4 – Estimate the model". Further details on how to apply a model using JSR 73: JDM compliant DME can be found in document [14].

6.4.6.4 InformMiningCompleted task

This task sends a message to the coordinator agent informing the end of miner agent processing. This task consists of one method: action().

6.4.6.4.1 action()

The algorithm of InformMiningCompleted task's action() method is shown in Algorithm 23.



Algorithm 23 Miner::InformMiningCompleted::action() method's algorithm

The message is prepared by setting the following parameters of the message:

- sender: the AID (agent identifier) of the miner agent
- receiver: the AID of the coordinator agent
- language: RDF is used in our implementation
- ontology: DCSOntology as described in section 5.1.1.

Then, the content of the message is filled with the "state" of the agent and the message is send.

7. CODE MODEL

This step consists of generating the code from the model using the PASSI add-in code generation functionalities and manual completion of the source code. Therefore, we will not include anything further in this section.

8. DEPLOYMENT MODEL

This section describes the distribution of the parts of the agent systems across hardware processing units and their migration between processing units and it involves the following phase:

- Deployment Configuration

The Deployment Configuration describes also any constraints on migration and mobility in addition to the allocation of agents to the available processing units.

8.1 Deployment Configuration

In this system, only one coordinator agent should exist and at least one data agent and one miner agent should exist. Moreover, only one data agent and miner agent should resides on each node. Those constraints are chosen for simplification purpose.

No migration or mobility is allowed in this earliest implementation of the system for simplification purpose, but eventually the mobility can be integrated easily since JADE support this feature.

In Figure 65, there are two nodes (Node 2 and Node 3) where data and miner agents reside to show that it is possible to have more than one data and miner agent each.

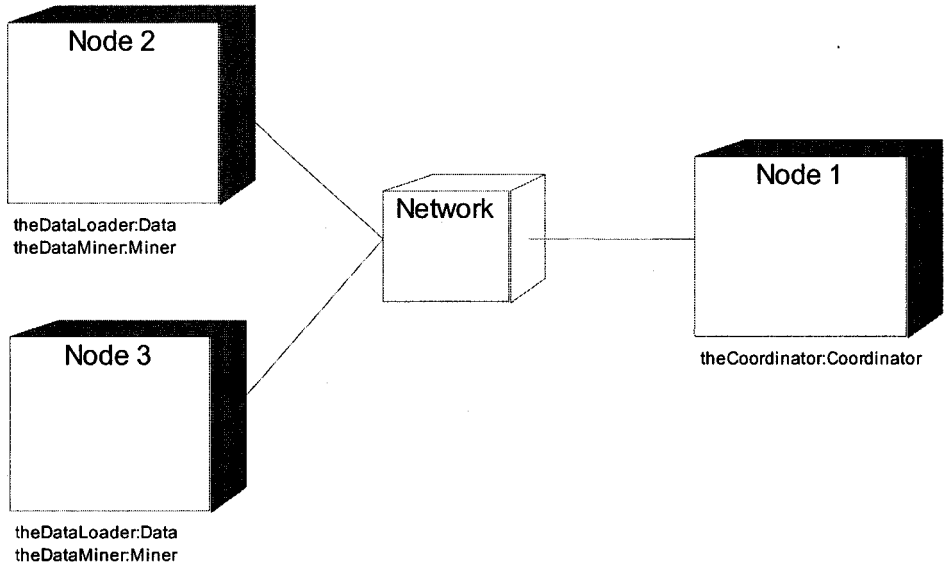


Figure 65 Deployment Configuration Diagram

APPENDIX 2

Use of unsupervised clustering algorithm in high dimensional dataset

USE OF UNSUPERVISED CLUSTERING ALGORITHM IN HIGH DIMENSIONAL DATASET

« UTILISATION DES ALGORITMES DE RÉSEAUX DE NEURONES DANS L'EXTRACTION DE CONNAISSANCES NON-SUPERVISÉE »

TECHNICAL REPORT

NESET SOZEN

Département de Génie Électrique

ÉCOLE DE TECHNOLOGIE SUPERIEURE
UNIVERSITE DU QUEBEC

MONTRÉAL, MAY 7, 2005

1. INTRODUCTION

This dissertation concerned with experimenting on high dimensional data several clustering methods such as Projective Adaptive Resonance Theory (PART) neural network and Fuzzy c-means (FCM) algorithm. These clustering methods will be used for knowledge extraction in context of *Knowledge Discovery* (KD).

KD is a process aiming at the extraction of previously unknown and implicit knowledge out of large databases which may potentially be of added value for some given application [1]. KD is achieved by using various data mining techniques such as classification, clustering, association, artificial neural networks, fuzzy logics, genetic algorithms, etc.

In practice, implementing a data analysis technology and using of DM methods efficiently requires highly qualified resources and it can be very costly to put into operation "in-house" data mining system or to subcontract the project to a third-party. Most of the time, the organization don't have the resources available to effort those options. Therefore, by using unsupervised data mining algorithms such as clustering algorithms, the external intervention (i.e. human interactions) to the process should be reduced to minimum.

The rest of this paper is organized as follow. In the next section, the concept of knowledge discovery and data mining are briefly covered. The data sets are described in section 3. In section 4, an overview of the selected approaches is given. Then, the experimental protocol and the results with an analysis are presented. Last section presents some further discussions and concluding remarks.

2. PROBLEM STATEMENT

The main goal of this study is to experiment some classical and specialized clustering methods and observe how they react in context of high dimensional space. We are not targeting to test or verify the algorithms itself but to verify their usability in our context. Hence, best responding methods in high dimensional space will be implemented in the Data Crawler System (DCS) for extracting knowledge from Bell's Operational Data Mart (ODM). The knowledge extraction is an elementary task of data mining and/or knowledge discovery.

Before going further in description of problematic, the following questions will be answered: what exactly are data mining and knowledge discovery? More fundamental, what is knowledge?

Data mining is an iterative process for discovering unknown knowledge from large volume of data by applying statistical and machine learning techniques. At high level, knowledge discovery and data mining process can be modeled as a process with inputs and outputs as in Figure 66.

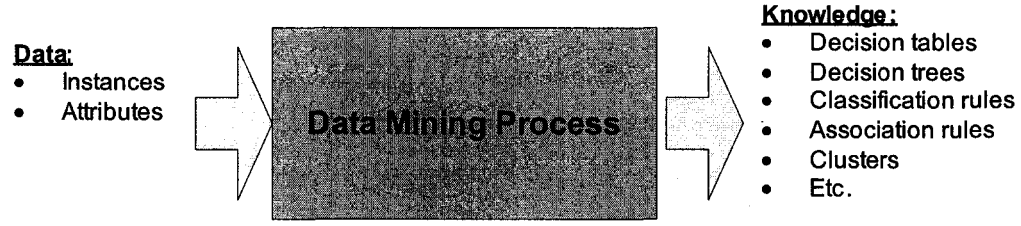


Figure 66 Data mining process's inputs and outputs

The inputs to data mining process are instances and attributes. The instances are thing that the DM techniques will be applied on and the attributes are characterization of each

instance. For example, each row of the database in Figure 67 corresponds to an instance and each column corresponds to an attribute.

ID	Age	Sex	Movie Type	Occupation
1	32	F	Suspense	Doctor
2	23	F	Adventure	Plumber
3	21	M	Action	Taxi driver
…	…	…	…	…

Figure 67 A database illustration

In [8], authors propose the concept another input to data mining process. They describe it as what will be learned. For example, if the data mining technique is classification, the expected outcome will be a set of classified instances and if the used technique is decision tree, the outcome will be an association rules between attributes and not a class. In spite of the learning scheme, what is learned is the concept.

These outcomes are *knowledge*. The knowledge is structural patterns and/or relations in data discovered by machine learning methods. It will have different representation depending on the used DM techniques. If the used technique is a decision tree construction algorithm the knowledge will be in form of decision tree and in case of clustering methods, the knowledge will be represented by clusters.

There are two approaches (or objectives) for using data mining: *prediction* and *description*. Prediction is about forecasting future data values. This type of data mining will produce model of the system based on the given data. The descriptive data mining will produce hidden knowledge patterns without a preset hypothesis about what the outcome may be. The goal is to gain an understanding of the system. With predictive approach, there is a question to answer, for example "what might be a good promotion for our new product?", "How much profit can be made for the next quarter?". With

descriptive approach there is a valuable data with no prior directive for what we are looking for.

There are many DM techniques, each of which falls into one of these following categories [3]:

1. *Classification* – discovery of a predictive learning function that classifies a data item into one of several predefined classes.
2. *Regression* – discovery of a predictive learning function, which maps a data item to a real-value prediction variable.
3. *Clustering* – a common descriptive task in which one seeks to identify a finite set of categories or clusters to describe the data.
4. *Summarization* – an additional descriptive task that involves methods for finding a compact description for a set (or subset) of data.
5. *Dependency Modeling* – finding a local model that describes significant dependencies between variables or between the values of a feature in a data set or in a part of a data set.
6. *Change and Deviation Detection* – discovering the most significant changes in the data set.

The steps of the knowledge discovery and data mining process, shown in Figure 68, are described below.

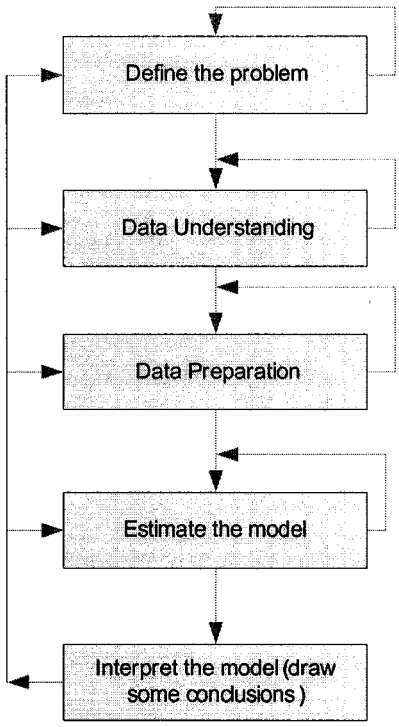


Figure 68 Data mining process

1. *Define the problem:* In this initial step a meaningful problem statement and the objectives of the project are established. Domain-specific knowledge and experience are usually necessary. In this step, a modeler usually specifies a set of variables for the unknown dependency and, if possible, a general form of this dependency as an initial hypothesis. There may be several hypotheses formulated for a single problem at this stage. The first step requires the combined expertise of an application domain and a data-mining model. [3]

2. *Data understanding:* This step is about data extraction and detection of "interesting" data subsets.

3. *Data preparation:* During this step the final dataset is constructed. Common data preparation tasks will be outlier detection and removal, scaling, encoding,

selecting features, etc. This step should be regarded as together with other data mining phases. Hence, a good data preparation method with a priori knowledge will provide optimal results from a data mining techniques.

4. *Estimate the model:* In this step, various data mining techniques and algorithms are applied on data set prepared previously. And the verification of the DM models to ensure that our model is robust and achieve the objectives specified during the problem definition.

5. *Interpret the model:* In this final step, the results are presented to client (decision maker). Since, the results are used to make decision; they should be understandable in form of simple reports using organization templates and not hundreds of meaningless numerical results.

Therefore, clustering algorithms, like many other statistical learning methods, can be used mainly at step 4 "estimate the model" of the data mining process.

In DCS project, there isn't any prior hypothesis or knowledge about that data that will be mined; therefore the selected approach will be descriptive. As a result, only unsupervised data mining methods will be used in earliest version of DCS. This is the main reason of studying clustering methods which is an unsupervised learning technique.

Subsequently, these clustering methods will be used to extract knowledge from Bell's ODM database. Each data entry in ODM database has near 300 features (dimensions). With such a high dimensional data sets, one needs to overcome many technical challenges.

A habitual approach to prevail over high dimensional data sets is *feature reduction* (or *feature transformation*) which consists of forming a new set (with less features) of feature from the original set using methods such as *Principle Component Analysis* (PCA). However, this approach can not be used in our project because we are doing descriptive data mining and lose of information, by transforming original data set, can not be allowed.

Traditional clustering methods do not perform efficiently with high dimensional data because of the *curse of dimensionality*. Curse of dimensionality is the fact that the demand for a large number of data samples grows exponentially with the dimensionality of the feature space. The fundamental reason for the curse of dimensionality is that high-dimensional functions have the potential to be much more complicated than low-dimensional ones, and that those complications are harder to discern. [31]

Recent theoretical results [27] have shown that in a high dimensional space, the distance between every pair of points is almost the same for a wide variety of data distributions and distances functions. As a result, clustering in original space of all dimensions will not give any conclusive results.

The concept of *subspace clustering* is motivated by inherent problems to curse of dimensionality described above. The goal of subspace clustering is to find clusters formed in subspaces of the original high dimensional space.

3. DATABASE AND FEATURE EXTRACTION

As mentioned earlier, eventually the studied clustering methods will be used on Bell's operational data. But we can not use directly these data sets to study selected methods because we don't have any prior knowledge about Bell's data. Consequently, there isn't any reference to validate the correctness of the produced clusters directly from Bell's data. Therefore, we will first use synthetic data, for carrying out the tests.

In this section, Bell's operational data and methods used to generate synthetic data will be described.

3.1 Bell's Operational Data

A data entry in ODM is composed of four types of attribute: numerical, date, string and nominal (which can be composed of character or number, i.e. (true/false) or (1/0)). The data set has near 300 attributes. The original data set sample that we get from Bell, which has been pruned by them, has 267 features including two key attributes and 24202 entries. After, removing all string and date type attributes and transforming all nominal attributes composed with characters and numbers, we had data set with 180 dimensions.

3.2 Synthetic Data

The high dimensional synthetic data is generated using the method proposed by Aggarwal et al. [50]. The data will be composed of numerical data only. At this level, string and nominal type will not be considered for simplification reasons.

The points have coordinates in the ranges [0 100] and are either cluster points or outliers. The $F_{outlier}$=5% is the percentage of data sets that are outliers and they are distributed uniformly at random throughout the entire space.

Synthetic data generator program takes as input parameters the number of clusters k and a Poisson parameter μ that determines the number of dimensions in each cluster, as shown in Algorithm 24.

1. Define anchor points around which the clusters will be distributed and the dimensions associated with each anchor point
2. Determine how many points will be associated to each cluster
3. Generates the cluster points

Algorithm 24 Synthetic data generator

3.2.1 Define anchor points

Anchor points are obtained by generating k uniformly distributed points in the d-dimensional space. The anchor point for the i^{th} cluster is denoted by c_i. The number of dimensions associated with a cluster is given by the realization of a Poisson random variable with mean μ. This number must be at least 2 and at most d. Once the number of dimensions d_i associated with cluster I is generated, the dimensions for each cluster are chosen using the following technique:

The dimensions for the first cluster are chosen randomly. The dimensions for the ith cluster are generated inductively by choosing $\min\left\{d_{i-1}, \frac{d_i}{2}\right\}$ dimensions from the (i-1)st cluster and generating the other dimensions randomly.

This iterative technique is proposed to model the fact that different clusters frequently share subsets of correlated dimensions.

3.2.2 Determine number of points

The number of points of each cluster is determined by generating k exponential random variables with mean 1 and by assigning to each cluster a number of points proportional to these realizations. More exactly, let $r_1, r_2, \ldots, r_k$ be the realizations of k random variables and let $N_c = N \cdot (1 - F_{outlier})$ be the number of cluster points. Then the number of points in cluster I is given by $Nc \cdot \frac{r_i}{\sum_{i=1}^{k} r_i}$.

3.2.3 Generate cluster points

The coordinates of the non-cluster dimensions are generated uniformly at random. For a cluster dimension j, the coordinates of the points projected onto dimension j follow a normal distribution with mean at the respective coordinate of anchor point and variance is determined randomly as follow:

> Fix a *spread parameter* r and choose a *scale factor* $s_{ij} \in [1,s]$ uniformly at random, where s is used defined. Then, the variance of the normal distribution on dimension j is $(s_{ij} \cdot r)^2$. In our case, r=s=2.

4. OVERVIEW CLUSTERING METHODS

We used three clustering methods: Projective Adaptive Resonance Theory (PART) algorithm, Fuzzy c-means (FCM) algorithm and k-means algorithm.

The paper [30] presents projective adaptive resonance theory (PART) neural network developed by Cao and Wu. PART architecture is a variation of adaptive resonance theory (ART). This method is revealed to be very effective in clustering data sets in high dimensional spaces. Fuzzy c-means is a popular clustering method proposed by J. C. Bezdek [29]. K-means is a classical statistical clustering method.

In this section each method will be described in details.

4.1 The PART algorithm

Traditional clustering methods do not work efficiently for high dimensional data sets because of the sparsity of data. Recent theoretical results [7] have shown that in high dimensional space, the distance between every pair of points is almost the same for a wide variety of data distributions and distance functions. Therefore, it makes no sense to talk about proximity or clustering on the original full space of all dimensions. We are in presence of curse of dimensionality. A solution to this problem will be to subspace clustering whose goal is to find clusters formed in subspaces of the original high dimensional space.

Projective Adaptive Resonance Theory (PART) offers a solution to the challenging problem of high-dimensional clustering problem. PART is a new neural network architecture similar to adaptive resonance theory (ART).

ART network has two key parts: choice process and match process. Choice process select the most likely cluster for an input pattern and if the chosen template cluster and the input pattern are similar enough according to a predefined vigilance parameter ρ, then the cluster's template is updated according to the new input pattern. Otherwise, the cluster is reset and the next most likely cluster is chosen. When no existing cluster satisfies the match criterion, a new cluster is created.

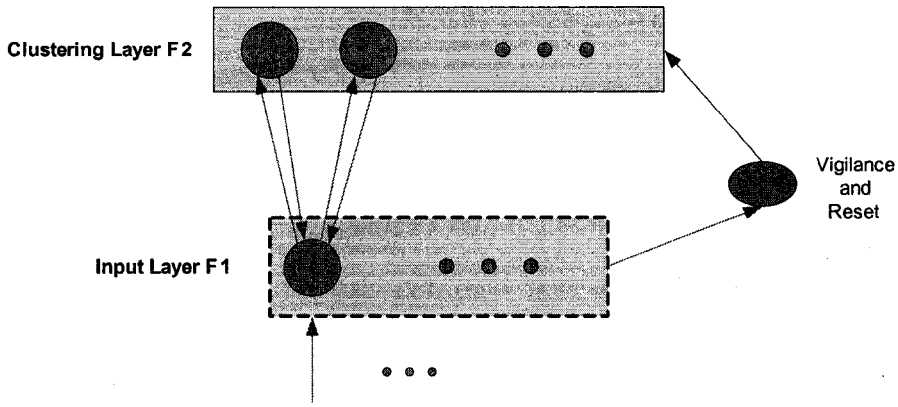


Figure 69 Simplified ART architecture

The ART architecture is illustrated in Figure 69. It has an input processing part (F_1 layer, also called comparison layer), a clustering part (F_2 layer) and a vigilance and reset subsystem. F_1 layer and F_2 layer are connected to each other by bottom-up weight and top-down weight respectively. F_2 layer follows winner-take-all paradigm and the node with the largest input becomes the candidate to learn the input pattern. Whether the candidate will learn the input pattern is decided by the vigilance and reset mechanism. The latter controls the degree of similarity of patterns placed in the same node (cluster).

The advantage of ART is that the number of cluster is not determined by the user, thus he can control the degree of similarity of patterns placed in the same cluster. ART network looks for similarity of patterns in the full dimensional space and sometimes fails to find patterns in the subspaces of higher dimensional space. Subspaces in which

clusters are formed can not be identified in advance because to find clusters in all possible subspaces and compare the results to get an optimal partition of data set, we have pass trough 2^m-1 subspaces with a high dimension of m which is practically unfeasible. To circumvent this problem a selective output signaling mechanism to ART is introduced. This change to ART will constitute the new PART architecture as illustrated in Figure 70.

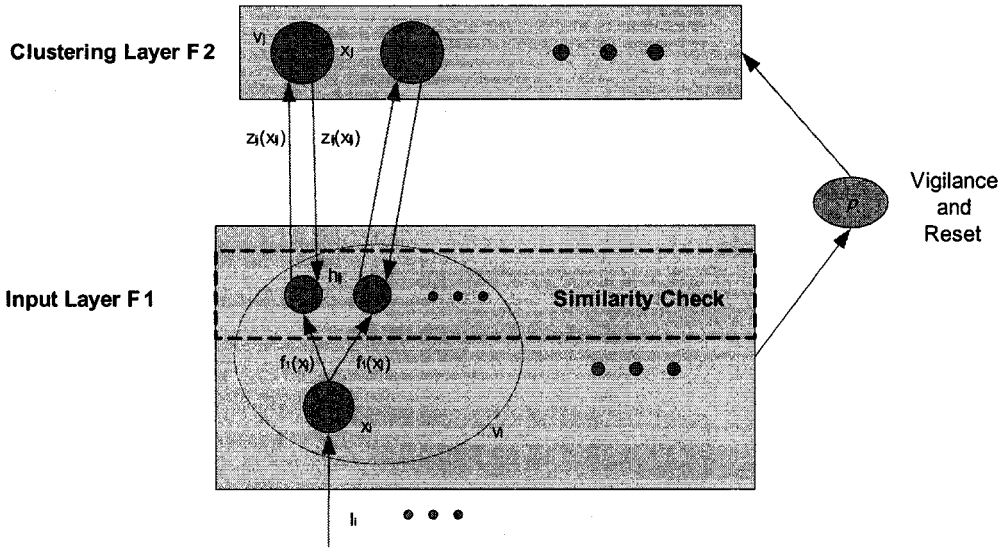


Figure 70 PART Architecture

where

v_i = *Node in* F_1 $i = 1,...,m$

v_j = *Node in* F_2 $j = m+1,...,m+n$

x_i = *Activation of* F_1 *node* v_i

x_j = *Activation of* F_2 *node* v_j

z_{ij} = *Bottom-up weight from* v_i *to* v_j

z_{ji} = *Top-down weight from* v_j *to* v_i

h_{ij} = *Selective output signal from* v_i *to* v_j *by a similarity check between* z_{ji} *and the signal* $f_1(x_i)$

f_1 = *Signal function*

F_2 layer of PART is similar to ART architecture; it is a competitive layer which follows the winner-take-all paradigm. The principal difference between ART and PART lies in the functioning of F_1 layer. In PART, F_1 layer selectively sends signals to nodes in F_2 layer. As a result, the node in F_1 is active relative to some F_2 nodes only and the activation is determined by a similarity check.

4.1.1 Detailed PART algorithm

The following are the input parameters of PART algorithm.

PARAMETER	PERMISSIBLE RANGE	SAMPLE VALUE
L (constant)	$L > 1$	2
α (learning rate)	$0 \le \alpha \le 1$	0.1
ζ (threshold)	$0 \le \zeta \le \frac{L}{L-1+m}$	0
ρ (vigilance parameter)		n/a
σ (distance vigilance parameter)		n/a

Detailed algorithm of PART neural network is as in Algorithm 25

0. Initialization:
   0.1. Take the number of nodes in F_1 layer as the number of dimensions of input data
   0.2. Choose the number n of nodes in F_2 layer is much larger than the expected number of clusters
   0.3. Set the internal parameters L, α and ζ and the maximum iteration M.
   0.4. Choose the external input parameters ρ and σ.

1. Set all F_2 nodes as being noncommitted

2. For each data point in input data set, do step 2.1-2.6.
   2.1. Compute h_{ij} for all F_1 nodes v_i and committed F_2 nodes v_j. If all F_2 nodes are noncommitted, go to Step 2.3.
   2.2. Compute T_j for all committed F_2 nodes v_j.
   2.3. Select the winning F_2 node v_J. If no F_2 node can be selected put the data point into outlier and then continue to do step 2.
   2.4. If the winner is a committed node, compute r_J, otherwise go to Step 2.6.
   2.5. If $r_J \geq \rho$, go to step 2.6, otherwise reset the winner v_J and go back to Step 2.3.
   2.6. Set the winner v_J as the committed, and update the bottom-up and top-down weights for winner node v_J.

3. Repeat step 2, M times.

4. For each cluster C_j in F_2 layer, compute the associated dimension set D_j.

Algorithm 25 Detailed PART neural network algorithm

Let $\Lambda_1 = \{1,\ldots,m\}$ denote the set of nodes in F_1 layer and $\Lambda_2 = \{m+1,\ldots,p\}$ denote the set of committed nodes in F_2 layer.

4.1.1.1 F_1 Activation and Computation of Selective Output Signals h_{ij}

The signal function f_1 will be the identical function $f_1(x_i) = x_i$ and at an equilibrium $x_i = I_i$ and consequently $f_1(x_i) = I_i$, where $I = \{I_1 \cdots I_m\}$ is an input pattern with m

dimension. Therefore, the selective output signal h_{ij} is computed by the following formula

$$h_{ij} = h_{\sigma}(d(f_1(x_i), z_{ji}))I_{\zeta}(z_{ij}) \quad t \geq 0 \quad i \in \Lambda_1 \quad j \in \Lambda_2$$

with σ being a distance vigilance parameter and ζ a small threshold usually taken as 0 and where $\sigma > 0$ and $0 \leq \zeta \leq \frac{L}{L-1+m}$ where L is a constant ($L > 1$)

and we take as d

$$d(a,b) = |a-b| \quad \text{for any} \quad a,b \in \square$$

and for a given constant c, h_c is given by

$$h_c(\xi) = \begin{cases} 1, & if \quad \xi \leq c \\ 0, & if \quad \xi \leq c \end{cases}$$

and I_c is given by

$$I_c(\xi) = 1 - h_c(\xi).$$

Then, h_{ij} will be

$$h_{ij} = \begin{cases} 1, & \text{if} \quad |f_1(x_i) - z_{ij}| \leq \sigma \quad \text{and} \quad z_{ij} > \zeta \\ 0, & \text{otherwise.} \end{cases}$$

4.1.1.2 F_2 Activation and Selection of Winner

The bottom-up filter input Tj to the committed F2 node vj is computed by

$$Tj = D\sum_{i \in \Lambda_1} z_{ij} h_{ij}$$

with $D = 1$ and the winner is selected according to the following rule

> *Let* $\Gamma = \{T_k : F_2$ *node* v_k *is committed and has not been reset on the current trial}, then node* v_J *is a winner either if* $\Gamma \neq \varnothing$ *and* $T_J = \max \Gamma$, *or if* $\Gamma = \phi$ *and node* v_J *is the next noncommitted node in F2 layer.*

4.1.1.3 Vigilance and Reset

A winning F_2 node v_J is reset if the matching degree r_J defined by

$$r_J = \sum_i h_{iJ}$$

is less than a vigilance parameter ρ. Specifically, reset occurs if and only if

$$r_J < \rho$$

where $\rho \in \{1, 2, \ldots, m\}$. Otherwise, the winner passes the vigilance test and is ready to learn the input pattern.

4.1.1.4 Learning

For the winner v_J which has passed the vigilance test, its bottom-up weights z_{iJ} and its top-down weights z_{Ji} are updated following the rules below:

If v_J is a committed winning F_2 node, then

$$z_{iJ}^{new} = \begin{cases} \dfrac{L}{L-1+|X|}, & if \quad h_{iJ} = 1 \\ 0, & if \quad h_{iJ} = 0 \end{cases}$$

$$z_{Ji}^{new} = (1-\alpha) z_{Ji}^{old} + \alpha I_i$$

where the learning rate is expressed by $0 \leq \alpha \leq 1$ and $|X|$ denotes the number of elements in the set $X = \{i : h_{iJ} = 1\}$. Therefore, $r_J = |X|$.

If v_J is a non-committed F_2 node, then

$$z_{iJ}^{new} = \frac{L}{L-1+m}$$
$$z_{Ji}^{new} = I_i$$

4.1.1.5 Dimensions of Projected Clusters

Each committed F_2 node v_j represents a projected cluster C_j. The set D_j of the associated dimensions of the projected cluster C_j is determined by $I_\zeta(z_{ij})$ according to the following formula:

The dimension $i \in D_j$ if and only if $I_\zeta(z_{ij}) = 1$.

4.1.1.6 Outlier Node

In the PART architecture, F_2 layer can have an unlimited number of nodes because it is not bound by a maximum value. Due to the practical resource restriction, the number of nodes in F_2 layer has to be limited. Besides, there are always some outlier data points that are not clustered properly when applying clustering methods. Therefore, a special node, called outlier node, will be added in PART module and all data that cannot be clustered properly into F2 nodes will be put into this outlier node.

4.1.2 Assumptions

The success of PART algorithm relies on the assumptions that the model equations of PART have a regular computational performance described by the following dynamical behaviors when a constant input is imposed during each learning trial:

- Winner-take-all paradigm: the F_2 node with the largest bottom-up filter input becomes the winner and only this node is activated
- Selective output signals remain to be constants
- Synaptic weights are updated following specific formula
- Set of dimensions of a specific projected cluster is non-increasing in time

4.2 Fuzzy c-means algorithm

Fuzzy c-means (FCM) is a procedure of clustering data wherein each data point belongs to a cluster to a certain degree that is specified by a membership grade. This technique was originally introduced by Jim Bezdek in 1981 [29] and in our experimentation, we will use the Matlab's implementation of fuzzy c-means.

4.2.1 Detailed Fuzzy c-means algorithm

This clustering method partitions a set of object $\{x_1,\ldots,x_n\} \subset \square^s$ into c-(fuzzy) clusters based on a computed minimizer of fuzzy within-group least squares functional

$$Jm(U,\upsilon) = \sum_{i=1}^{c}\sum_{k=1}^{n} U_{ik}^{m} \left\| \mathrm{x}_k - v_i \right\|_2^2$$

where

$m > 1$	Fuzzification parameter
$v_i \in \square^s$	Prototype (or mean) of the i^{th} cluster
$U_{ik} \in [0,1]$	Degree to which datum x_k belongs to the i^{th} cluster

$\upsilon = \left[\upsilon_{ji}\right] = \left[v_1, \ldots, v_c\right] \in \square^{s \times c}$	Matrix of cluster prototypes
$U = \left[U_{ik}\right]$	Partition matrix
$\|*\|_2^2$	Euclidean or 2-norm squared

The set of all nondegenerate fuzzy $c \times n$ partition matrices for partitioning n data into c clusters as

$$M_{fcn} = \left\{ U \in \square^{c \times n} \mid \forall i,k : 0 \le U_{ik} \le 1; \sum_{i=1}^{c} U_{ik} = 1; 0 < \sum_{k=1}^{n} U_{ik} \right\}$$

The most popular and effective method of optimizing the equation *Jm(U, υ)* is fuzzy c-mean algorithm, which alternates between optimizations of $\mathcal{J}_m\left(U \mid \upsilon^*\right)$ over U with υ^* fixed and $\mathcal{J}_m\left(\upsilon \mid U^*\right)$ over υ with U^* fixed, producing a sequence $\left\{\left(U^{(r)}, \upsilon^{(r)}\right)\right\}$. Specifically, the $r+1$ st value of $\upsilon = \left[v_1, \ldots, v_c\right]$ is computed using the rth value of U in the right-hand side of

$$v_i = \frac{\sum_{k=1}^{n} U_{ik}^m \mathrm{x}_k}{\sum_{k=1}^{n} U_{ik}^m}, \quad \text{for} \quad i = 1, \ldots, c$$

Then the updated $r+1$ st value of υ is used to calculate the $r+1$ st value of U via

$$U_{ik} = \frac{d_{ik}^{\frac{-1}{(m-1)}}}{\sum_{j=1}^{c} d_{jk}^{\frac{-1}{(m-1)}}}$$

where $d_{ik} = \left\|\mathrm{x}_k - v_i\right\|_2^2 > 0 \quad \text{for} \quad i = 1, \ldots, c \quad \text{and} \quad k = 1, \ldots, n$

The FCM iteration is initialized using some $U \in M_{fcn}$ (or possibly $\upsilon \in \square^{s \times c}$) and continues by alternating the updates in equation of v_i and U_{ik} above until the difference measured in any norm on $\square^{c \times n}$ (or $\square^{s \times n}$) in successive partition matrices (or υ matrices) is less than some prescribed tolerance ε.

4.3 K-means algorithm

This algorithm is based on the minimization of the performance index (J) defined by the sum of the squared distances of all points of a cluster to its mean value:

$$J = \sum_{j=1}^{N_C} \sum_{X \in S_j} \left\| X - m_j \right\|^2$$

where N_C is number of points in the cluster's domain,
X is the point to be classified
m is the mean value

The algorithm will have as input parameter the number of clusters and the means value of each clusters. The algorithm of k-means clustering [31] will be as follow

Algorithm 1 (K-means clustering)

1 **begin initialize** n, c,$\mu 1$,$\mu 2$, . . . ,μc

2 **do** classify n samples according to nearest μi

3 recompute μi

4 **until** no change in μi

5 **return** $\mu 1$,$\mu 2$, . . . ,μc

6 end

Algorithm 26 K-means clustering algorithm

where n is the data sample to classify, c is the number of cluster and the μ is mean value. Further detail on this method can be found in [31] or any classification pattern book.

5. EXPERIMENTAL PROTOCOL

The experimental protocol is as follow:

- *Analyze data sets from Bell's ODM and generate similar synthetic data:*

  As mentioned earlier, our goal in doing this study is to determine a strategy to find out the most appropriate data mining methods to analyze data from Bell's ODM. The most distinctive aspect of the ODM is the high dimensionality. Also, the selected approach (descriptive data mining) in extracting knowledge has a huge impact on the type of methods too.

  There are several obstacles to overcome. First, we don't have any prior knowledge about the data set and we should consider all possible structures and correlations in data. The original data can't be used to experiment because it isn't labeled. A data instance is labeled when we know which class it is associated with and in our case we have no clues even about the possible classes (or clusters) inherent in data set. Often in high dimensional data, many dimensions are irrelevant and mask existing clusters and the clusters aren't in full dimension. Another obstacle that the clustering algorithms have to face is the curse of dimensionality. The subspace clustering theory arises as a solution to cluster high dimensional data.

  To simulate the Bell's ODM, synthetic data set with clusters in different subspace will be generated using the method described in section 3.2. The data set will have 300 dimensions and 30000 entries. The value of each input will be [0,100]. Since we don't have any prior knowledge, we will modify the size of the subspace (number of dimensions) of each

cluster in a data set and observe the ability of the methods to discover the clusters. Three types of data set will be generated:

- Clusters with high-dimensional subspace (between [250, 300[dimensions)
- Clusters with low-dimensional subspace (less than 15 dimensions)
- Clusters within full range subspace (between [2,300[dimensions)

- *Implement PART algorithm:*

  PART algorithm will be implemented using Java language. For others two methods (fuzzy c-means and k-means), their Matlab implementation will be used directly in our experimentation.

  The PART algorithm was first implemented in Matlab and worked well on data with few dimensions but when we simulated it with higher dimensions (> 100) we couldn't get any results due to resource restriction with Matlab and long simulation time. We had memory overload error with only 5000 entries. Therefore, we decided to implement the PART algorithm using Java, where we have much more control over resources than Matlab. Also, we can directly use the PART implementation later during the implementation of the data crawler system.

- *Execute clustering algorithms on synthetic data and compare results:*

  With PART algorithm, a prior analysis is executed in order to determine the external parameters σ and ρ. We took the same value as in document [30] for parameter σ because we used the same range of value for each data entry. To determine the value of ρ, a prior execution with one

iteration only of the algorithm on each data set with ρ value varying from 2 to 15 was performed.

All algorithms are compared using a contingency table of input clusters and output clusters as in Figure 71.

Input / Output	Clusters 1	2	...	Outlier	Sum
1					
2					
...					
outlier					
Sum					

Figure 71 Contingency table

Each entry *(i,j)* denotes the number of data points that are common to both output cluster *i* and the input cluster *j*.

6. EXPERIMENTAL RESULTS

In this section, the experimental results are given and analyzed. The results are reported with emphasis on the contingency table of input clusters and output clusters. The complete results are annexed at the end of this document (section 9).

6.1 PART algorithm

As show in document [30], PART is a subspace clustering method effective in high dimensional data sets and we would like use it for analyzing data from ODM. In order to use this algorithm effectively, we need to set some external parameters: σ and ρ. The σ will be set to 10 as we are using same range of value for each point in synthetic data as in [30]. The vigilance parameter ρ is directly related to the number of dimension on which the cluster is defined. For example, ρ=1 means that two points are close to each other with respect to at least 1 dimension. Therefore, a prior test is executed to determine the most appropriate ρ value. For example, for high dimensional subspace clusters data set (~270-dimensional clusters), the best results are get with ρ=13 and for low-dimensional subspace clusters data set (~10-dimensional clusters), the best results are get with ρ=6 and in case of the data set with clusters dimensions varying within full range subspace ρ=3 gave the best results.

With the three data sets we studied the capability of PART algorithm for finding the clusters depending on their dimensional size.

Clusters with high-dimensional subspace:

With ρ = 13, PART algorithm was successful in finding the exact number of clusters with their exact entries. Even, all of the outliers are detected properly (1498 on 1500 are detected). However, with ρ = 6, only 7 clusters were found where 4 of them were input

clusters and other 3 were combination of the input clusters. The 7 found clusters are as in Table XVI.

Table XVI

Clusters with high-dimensional subspace

Output Cluster	**Input cluster**
1	1
2	2
3	3-4
4	5-6
5	7-8
6	9
7	10

Only input clusters 1, 2, 9 and 10 are found properly.

Clusters with low-dimensional subspace:

The PART algorithm is applied on this data set with $\rho = 6$ and $\rho = 3$. With $\rho = 6$, no clusters were detected because the dimensional size of the clusters was too small and the vigilance parameter of 6 was too high. The vigilance parameter indicates the number of dimensions on which the distance between each point is evaluated.

With $\rho = 3$, we were able to detect input clusters 2, 3, 4, 5, 6, 7, 8 and 9. The input clusters 1 and 10 could not be found because their dimensional size is 2 (which is less than 3). Usually, there isn't correlation in large number of random data points with large set of dimensions. Therefore, we couldn't use vigilance parameter value less than 3 because we need to select a ρ value high enough to eliminate the randomness in data set.

Another aspect to discuss is how accurate the input clusters are discovered by the PART algorithm. The accuracy is computed as follow

$$accuracy = 100\% - \left(\frac{x-y}{x}\right)$$

where x= size of the output cluster
y= size of the input cluster

Table XVII

Clusters with low-dimensional subspace

Output Cluster	**Accuracy (%)**
2	59%
3	63%
4	62%
5	66%
6	78%
7	66%
8	77%
9	40%

Compared to results obtained with clusters with high-dimensional subspace with $\rho = 13$ where we had output clusters with near 100% accuracy, the highest accuracy is 78%. This is due the fact that the vigilance parameter of $\rho = 3$ isn't high enough to eliminate the randomness.

Clusters within full range subspace:

With this test, we wanted to study the ability of PART algorithm to discover clusters with different dimensional size in the same data set. In this data set, we have clusters

with 10-dimensional subspace as well as 275-dimensional subspace. This test is conducted with vigilance parameter of $\rho = 13$ and $\rho = 6$.

With $\rho = 13$, 8 clusters were found. These founded clusters correspond to the 8 inputs clusters (1 to 8). PART algorithm considered data points from input clusters 9 and 10 as outliers and these input clusters could not be detected. The fact that the input cluster 9 is considered as an outlier is expected because this input cluster has 10-dimensional subspace and 10 is less than the vigilance parameter value of 13, which means that two points are considered close to each others with at least 13 dimensions. The input cluster 10 also could not be found even it has 30-dimensional subspace which is much higher than 13.

With $\rho = 6$, 9 clusters were found. All input clusters were found except the input clusters 7 and 8 were considered as one clusters. Both clusters were defined on high-dimensional subspace (input cluster 7 is defined on 250 dimensions and clusters 8 is defined on 275 dimensions) but the input cluster 7 had only 651 data points which correspond to ~2% of total number of data points. We also computed the distance of the centers of each cluster and we discovered that for 57 dimensions the distance value is less than 10 which is the value of the distance vigilance parameter σ. Therefore, low number of data points of input cluster 7 and the closeness of the center point of two input clusters could explain this combination.

6.2 Fuzzy c-means algorithm

A slightly different strategy was adopted for this algorithm because with PART algorithm we knew that it is designed and proven to be efficient for high dimensional data with subspace clusters [30] but it is not the case for fuzzy c-means. Therefore, instead of applying FCM directly on high-dimensional data sets(300 dimensions), we progressively increased the number of dimensions of data sets and observed if the FCM

was still able to detect clusters. The number of clusters in input data sets was 10 and number of input data was 10000 and we increased dimension of data set as follow: 5, 10, 20, 30, 50 and 75. We stopped at 75 because there was no change in the ability of FCM to detect clusters. All results are in the section 9.2.

The following table represents all dimensions of the input clusters with 5-dimensional data sets. First column represent the cluster id and the second all dimensions of the clusters.

Table XVIII

Input clusters with Fuzzy c-means

Clusters	dimensions
1	1,2,3,4
2	1,2
3	1,2,3
4	1,2,3
5	1,2,3,4
6	1,2,3,4,5
7	1,2
8	1,2,3,4
9	1,2,3,4
10	1,2

The following is a contingency table of input clusters and output clusters. As it can be observed in Table XIX input clusters 5 and 6 are clustered properly while the rest of clusters aren't. Cluster 5 is found with 16.5% (1763-1472/1763) of error while cluster 6 with 9% of error.

Table XIX

5-dimensional data sets

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	0	24	0	88	0	0	0	0	581	178	38	909
2	98	53	0	0	0	0	34	0	0	83	53	321
3	63	5	0	123	1472	0	20	0	0	0	80	1763
4	0	33	26	0	0	0	0	0	0	473	47	579
5	11	18	59	28	0	0	3	0	560	110	42	831
6	10	32	22	15	0	0	29	0	702	255	46	1111
7	54	0	49	0	0	0	16	0	0	398	36	553
8	0	50	0	113	0	0	17	0	746	439	100	1465
9	0	0	9	0	0	1191	0	0	0	100	9	1309
10	22	0	23	0	0	0	310	571	0	184	49	1159
Sum	258	215	188	367	1472	1191	429	571	2589	2220	500	10000

It also worth to mention that input cluster 6 is defined on full-dimensional space and only 4 of 5 dimensions are used to define cluster 5. Other input clusters defined using only 4 dimensions such as input clusters 1, 8 and 9 could not be found. The clusters need to be defined in full-dimension for FCM to identify them.

In 10 dimensional and higher data sets none of input clusters are properly classified. Therefore, we can say that FCM isn't suitable for extracting subspace clusters. Since, even with only 5 features, FCM is only able to extract 20% (2 of 10 input clusters) of the input clusters and the clusters must be defined using all dimensions. We didn't experiment the impact of the number of input data set because of the poor results with dimensionality.

6.3 K-means Algorithm

With k-means algorithm was executed only with two data sets of 5 dimensions; in the first data set the clusters were defined on subspace with different dimensional size and in the second data set all clusters were defined using all dimensions. Each data set had 10000 data points and 10 clusters. Only 5 dimensional data set were used because it was enough to show that k-means isn't suitable for clustering data with subspace clusters.

In case of data set formed with different dimensional size clusters none of the input clusters were identified while with data set containing full dimensional clusters all input clusters were discovered excepting input clusters 3 and 5 were found as one cluster. Detailed results are in annex 0.

7. DISCUSSION

As we can observe from our results, most of the classical clustering methods aren't efficient for mining data from the Bell's ODM because of the sparsity of data in high dimensional space. Usually, for two points from such data sets, there will be few dimensions on which the points are far apart from each other. As seen with k-means and fuzzy c-means results, searching clusters with traditional clustering methods in full dimensions can lead to erroneous results. Subspace clustering methods are an excellent way of mining high dimensional data.

Another important aspect to take into account is the input parameters of clustering methods. For example, k-means and fuzzy c-means need the number of searched clusters to be specified. In our case we don't have any prior knowledge and we are doing descriptive data mining, thus we don't have the slightest idea about the number of searched clusters.

Besides knowing that in our data crawler system, we can't use traditional clustering methods such as k-means and fuzzy c-means and we should decide on subspace clustering methods, another important output of this study is the establishment of a strategy for selecting data mining methods for our system. The following criteria should be considered in selecting a method for our system:

- *Prior knowledge requirements and domain specific strategy of DM methods:*

  Prior knowledge requirements are the most important aspect to consider for choosing a method for data mining because of our lack of knowledge about data. For example, if the method to be selected requires any knowledge related to clusters such as number of clusters or number of dimensions (or the specific dimensions) that form the clusters or any other information specifying the form of the clusters, the method should be rejected.

Otherwise, the input parameters of the method should be identified and the value of the input parameters should be defined according to our data. The domain specific strategy of data mining methods is the establishment of a strategy for defining the prior knowledge requirements of data mining methods according to input data space. For example, for the PART algorithm we need to define two external parameters (vigilance parameter (ρ) and distance vigilance parameter (σ)) that control the size of dimensions of the projected subspaces and the degree of similarity in a specific dimension involved respectively.

Usually, there is no correlation in large number of random data points with large set of dimensions. We observed that if we use smaller value for ρ, we are able to discover clusters with few dimensions (number of dimension is higher ρ value) but the cluster with many dimensions couldn't be extracted because of the inherent randomness in data. For example, in case of data set with clusters within full range dimensional subspace (see results in section 9.1.3) with ρ=6, we been able to discover all input clusters except the input clusters 7 and 8 which had high dimensional size (input cluster 7 was formed by 250 dimensions and input cluster 8 was formed by 275 dimensions). Therefore, we should choose ρ large enough to eliminate the randomness, but smaller than or equal to the number of dimensions of any possible projected cluster.

The strategy for selecting the value of ρ will be as follow:

PART algorithm will be executed with four values of ρ = [3, 6, 9, 12]. As a result, four instances of PART algorithm will mine the same data set.

In our experimentation the value of σ was 10 which corresponds to variation of ±10% (10/100 where 10 is the value of σ and 100 is the difference between maximum and minimum value (100-0)) of the current data entry from the compared cluster (F2 node). Therefore, if the difference between the current data entry and the F2 node was less than 10% than the current data input was considered as being part of the F2 node (for more details see paragraph 4.1 or document [30]).

The strategy for selecting the value of σ will be as follow:

$$\sigma_i = 0.10\left|\max_i - \min_i\right| \quad \text{where} \quad i = 1,\ldots,m$$

In our experimentation, the value of σ was equal for all dimensions because all dimension had the same range of value but in the implementation, there will be a different value for each dimension *i* because each dimension doesn't necessarily have the same range of value. The value of 10% of total range is chosen arbitrarily based on our experimentation.

- *Unsupervised, Descriptive data mining methods::*

  The goal in building the data crawler system is to gain an understanding of Bell's operations by uncovering patterns and relationships in ODM. In literature, this is know as descriptive data mining which produces new, nontrivial information based on the available data set.

  Unsupervised learning's goal is to discover "natural" structure in the input data and there is no notion of output during the learning process compared to supervised learning where unknown dependencies are estimated from known input-output samples. For example, with classification analysis which is a supervised learning, the output is taken

from an set of known class but with clustering which is an unsupervised learning, we don't have a set of a priori known clusters.

Therefore, only unsupervised descriptive data mining methodologies such as clustering analysis, association rules, or some artificial neural networks such as PART algorithm will be used.

- *Sensibility to high dimensionality:*

  As shown in [27], in a high dimensional space, the distance between every pair of points is almost the same for a broad variety of data distributions and distance functions. For example, in such condition, we can't perform clustering in full space of all dimensions. The subspace clustering which aims to find clusters formed in subspaces of the original high dimensional space is a viable solution to the problem.

  Therefore, we will opt for data mining methods that are not affected by the high dimensionality of data.

- *Scalability:*

  Scalability of data mining methods isn't a priority requirement because our system isn't online and the quality of the extracted knowledge is more important than the responsiveness of the system. However, it is an aspect to consider in design because there exist some methods which offer a good balance between performance and output quality such as MAFIA [28] clustering method. This algorithm uses an adaptive grid based on the distribution of data to improve efficiency and cluster quality and introduces parallelism to improve scalability.

- *Complexity (processing time):*

Same as with scalability, the complexity isn't a priority requirement but it will be considered in design and during the selection of the data mining methods the same way as scalability.

8. CONCLUSION

This project was a good opportunity to understand the concept of subspace clustering and the impact of large set of dimensions on output clusters. We observed that even with only with 5-dimensional data sets traditional clustering methods wasn't able to find most of the initial subspace clusters. Therefore, subspace clustering methods which are very efficient in finding clusters in different subspaces within a dataset, are preferred, against classical clustering methods, to be used in data crawler system.

Unfortunately, during the implementation we faced up some obstacles. The first Matlab implementation of PART neural network algorithm which behaved properly with low dimensional and with fewer data sets but with higher dimensional size (>100 dimensions) and with number of instances more than 5000, we couldn't get any results because of memory overload. Therefore, we decided to implement PART algorithm with Java programming language, which offers more flexibility in memory management. Also, the simulation time was considerably reduced and the source code in Java can be used later in the data crawler system implementation.

Our experimental procedure had some minor deficiency. For example, random entry for each data input is better experimental practice than use of each data input sequentially as we did. Also, we should compare the input-output clusters in basis of their dimensions and their anchor point (center point). However, as mentioned earlier, our goal wasn't to test the methods itself but to study their possible use in our data crawler system.

As shown in article [51], there are many other subspace clustering methods such as MAFIA, CLTree, Cell-based Clustering Method (CBF), which don't require prior knowledge related to the output cluster's form like PART algorithm. It will be interesting to do further studying for implementing those methods in the data crawler system.

9. ALL EXPERIMENTAL RESULTS

9.1 PART algorithm results

We had three types of data sets: data set with high-dimensional subspace clusters, data with low-dimensional subspace clusters and data sets with cluster's subspace vary within full range of dimension. The algorithm parameters are set as follow

PARAMETER	VALUE
L (constant)	2
α *(learning rate)*	0.1
ζ *(threshold)*	0
ρ *(vigilance parameter)*	13-6-3
σ *(distance vigilance parameter)*	10
M (number of iteration)	25

In order to make the results representation easier, clusters with less than 300 entries was considered as an outlier, which corresponds to 1% of total number (30000).

9.1.1 Clusters with High-dimensional Subspace

The input clusters are as follow:

Table XX

Input Clusters

Clusters	Size	Dimensions	Points
1	289	All dimensions except for 33, 39, 54, 89, 102, 122, 129, 159, 209, 293, 297	715
2	261	All dimensions except for 33, 57, 68, 82, 95, 97, 106, 118, 123, 129, 151, 161, 171, 177, 183, 189, 200, 201, 211, 219, 228, 232, 233, 237, 238, 239, 242, 244, 250, 252, 253, 257, 260, 263, 275, 280, 285, 298, 300	1970
3	281	All dimensions except for 12, 42, 131, 137, 143, 162, 166, 172, 175, 177, 210, 228, 242, 252, 264, 276, 283, 292, 298	2704
4	284	All dimensions except for 56, 107, 145, 148, 151, 171, 190, 202, 221, 232, 247, 251, 255, 258, 274, 284	1186
5	282	All dimensions except for 72, 85, 91, 126, 158, 168, 180, 183, 186, 191, 229, 243, 256, 260, 269, 280, 283, 291	1515
6	249	All dimensions except for 36, 51, 54, 73, 92, 104, 112, 129, 147, 152, 158, 161, 163, 164, 168, 169, 172, 178, 181, 184, 186, 194, 203, 204, 205, 206, 208, 210, 212, 213, 217, 220, 221, 226, 227, 229, 236, 241, 246, 248, 249, 256, 264, 265, 271, 274, 275, 285, 290, 291, 300	777
7	290	All dimensions except for 64, 131, 140, 141, 155, 171, 176, 242, 246, 283	2133
8	270	All dimensions except for 88, 91, 101, 106, 116, 125, 142, 153, 156, 166, 168, 174, 190, 191, 196, 237, 242, 247, 251, 254, 261, 262, 263, 268, 269, 272, 289, 290, 293, 296	3384
9	288	All dimensions except for 104, 113, 121, 149, 176, 208, 211, 244, 254, 275, 277, 285	3723
10	288	All dimensions except for 104, 138, 183, 195, 199, 203, 211, 235, 242, 278, 290, 296	10393

The following results are obtained when the external parameter $\rho = 13$.

Table XXI

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	714	0	0	0	0	0	0	0	0	0	0	714
2	0	1970	0	0	0	0	0	0	0	0	0	1970
3	0	0	2704	0	0	0	0	0	0	0	0	2704
4	0	0	0	1186	0	0	0	0	0	0	0	1186
5	0	0	0	0	1515	0	0	0	0	0	0	1515
6	0	0	0	0	0	777	0	0	0	0	0	777
7	0	0	0	0	0	0	2133	0	0	0	0	2133
8	0	0	0	0	0	0	0	3384	0	0	0	3384
9	0	0	0	0	0	0	0	0	3723	0	1	3724
10	0	0	0	0	0	0	0	0	0	10393	1	10394
outlier	1	0	0	0	0	0	0	0	0	0	1498	1499
Sum	715	1970	2704	1186	1515	777	2133	3384	3723	10393	1500	30000

The following results are obtained when the external parameter $\rho = 6$.

Table XXII

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	715	0	0	0	0	0	0	0	0	0	0	715
2	0	1970	0	0	0	0	0	0	0	0	0	1970
3	0	0	2594	1186	0	0	0	0	0	0	0	3780
4	0	0	0	0	1447	777	0	0	0	0	1	2225
5	0	0	0	0	0	0	2119	3384	0	0	0	5503
6	0	0	0	0	0	0	0	0	3723	0	0	3723
7	0	0	0	0	0	0	0	0	0	10393	3	10396
outlier	0	0	110	0	68	0	14	0	0	0	1496	1688
Sum	715	1970	2704	1186	1515	777	2133	3384	3723	10393	1500	30000

9.1.2 Clusters with Low-dimensional subspace

The input clusters are as follow:

Table XXIII

Input Clusters

Clusters	Size	Dimensions	Points
1	2	116,119	845
2	9	225,183,125,88,181,115,81,213,77,	3460
3	4	265,171,80,40	682
4	6	115,103,80,59,71,299	2251
5	7	78,103,85,106,126,167,240	3309
6	4	78,55,85,106	6407
7	11	114,55,63,106,155,225,50,88,97,249,52	1796
8	7	86,55,91,194,239,65,50	4043
9	6	86,55,39,7,134,65	1140
10	2	150,55	4566

The following results are obtained when the external parameters $\rho = 6$. When this test was executed, we obtained 100 clusters (which was the maximum number of clusters that we set as an input to limit used resources) where the cluster that had the highest number of instances was 11. Therefore, to facilitate the readability of the contingency table, all these clusters were considered as outliers.

Table XXIV

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
outlier	845	3460	682	2251	3309	6407	1796	4043	1140	4566	1500	29999
Sum	845	3460	682	2251	3309	6407	1796	4043	1140	4566	1500	29999

The following results are obtained when the external parameter $\rho = 3$.

Table XXV

Contingency table

Input / Output	Clusters										Outlier	Sum
	1	2	3	4	5	6	7	8	9	10		
1	3	2041	0	0	0	0	1	0	0	2	4	2051
2	2	12	428	0	0	0	0	0	0	1	9	452
3	1	0	0	1403	0	0	0	2	0	1	9	1416
4	5	12	6	6	2195	2	0	2	0	3	7	2238
5	1	17	5	10	0	4976	0	0	0	2	15	5026
6	11	15	5	17	20	43	1186	0	0	2	8	1307
7	3	22	5	15	21	68	8	3113	0	0	4	3259
8	3	19	2	9	12	0	0	0	456	6	2	509
9	7	25	4	20	22	213	34	0	0	0	9	334
10	4	7	5	11	15	0	0	0	33	367	2	444
11	1	14	6	18	16	0	0	0	28	512	2	597
12	1	16	4	12	10	0	0	0	33	649	9	734
13	4	27	3	13	30	0	0	0	0	741	18	836
outlier	799	1233	209	717	968	1105	567	926	590	2280	1402	10796
Sum	845	3460	682	2251	3309	6407	1796	4043	1140	4566	1500	29999

9.1.3 Clusters within full range subspaces

The input clusters are as follow:

Table XXVI

Input Clusters

Clusters	Size	dimensions	Points
1	15	4,97,102,104,114,138,182,189,200,206,210,217,218,271,295	961
2	100	2,4,12,13,16,19,20,24,27,37,45,47,48,53,54,56,57,58,60,62,63, 64,65,66,68,70,76,81,85,86,94,95,97,98,100,102,104,106,108, 114,118,119,122,123,125,129,140,147,150,151,158,167,174,175, 180,183,187,188,194,195,197,198,199,200,201,203,204,205,206, 207,210,213,217,221,224,226,227,228,231,233,236,239,242,244, 247,251,253,257,264,266,268,269,272,275,278,280,285,294,295,298	2217
3	150	1,2,3,4,5,6,8,9,10,11,12,13,16,17,19,20,21,23,24,27,29,31,32,35,36,37, 39,42,43,45,46,47,48,49,50,51,52,53,54,56,57,58,60,63,64,66,68,71,76, 77,80,81,82,83,84,85,86,88,89,90,92,93,97,98,99,102,104,105,107,108, 112,114,115,116,119,122,124,125,127,128,129,131,134,136,137,140, 141,142,144,147,148,150,151,152,153,159,162,164,166,168,170,174, 175,176,178,182,184,185,187,188,190,191,194,197,200,203,205,206, 208,212,213,217,219,220,222,225,228,232,234,237,238,242,244,251, 253,254,255,256,260,267,273,278,279,280,285,287,288,289,290,295	1288
4	200	1,2,3,4,5,6,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,45,46,47,48,49,50,51, 52,53,54,56,57,58,60,61,62,63,64,65,66,67,68,69,70,71,73,74,75,76,77, 78,79,80,82,83,85,86,88,89,90,92,93,94,96,98,101,102,107,108,109,110, 111,113,114,116,117,120,121,123,124,127,130,132,135,136,137,140,142, 144,146,149,150,151,156,157,160,161,162,164,165,167,170,171,172,173, 175,178,181,182,183,184,185,187,188,189,192,194,195,197,201,202,203, 207,209,210,211,213,214,216,217,220,221,222,225,227,230,233,234,237, 238,240,242,243,244,245,248,254,255,256,258,261,262,264,265,266,267, 268,269,271,273,274,276,278,280,282,283,284,285,286,287,288,289,291, 292,293,295,296,299	5388

Table XXVII

Input Clusters (cont.)

Clusters	Size	dimensions	Points
5	50	1,2,3,6,7,9,10,11,12,14,17,19,22,27,29,31,32,34,35,41,43,45,47,49,51,52, 56,57,59,60,66,68,71,75,81,84,90,92,93,121,128,133,191,199,207,225, 236,260,285,296	3810
6	75	1,2,3,6,7,9,10,11,12,14,16,17,21,22,27,29,31,32,34,38,41,43,44,45, 47,49,51,53,55,56,59,60,66,67,68,70,71,72,75,76,78,79,80,81,82,84, 89,99,112,117,121,125,126,128,129,143,146,152,165,168,189,199, 214,227,228,241,245,246,247,248,256,271,273,284,300	4679
7	250	All dimensions except for 50, 54, 55, 56, 57, 62, 66, 70, 86, 88, 93, 95, 99, 104, 119, 120, 121, 123, 124, 126, 133, 150, 151, 159, 179, 180, 182, 186, 192, 194, 196, 213, 215, 219, 223, 224, 234, 240, 249, 258, 262, 263, 267, 269, 272, 273, 274, 277, 287, 300	651
8	275	All dimensions except for 50, 59, 88, 99, 150, 169, 178, 180, 185, 202, 208, 214, 215, 218, 219, 224, 233, 236, 244, 251, 255, 270, 274, 280, 294	4255
9	10	2,10,21,22,31,58,79,110,129,173	1624
10	30	2,7,10,20,21,31,59,73,79,97,106,110,114,122,129,138,154,162,182, 190,207,217,228,229,238,239,243,244,257,284	3628

The following results are obtained when the external parameter $\rho = 13$.

Table XXVIII

Contingency table

Input / Output	Clusters										Outlier	Sum
	1	2	3	4	5	6	7	8	9	10		
1	929	0	0	0	0	0	0	0	0	0	0	929
2	0	2216	0	0	0	0	0	0	1	0	0	2217
3	0	0	1288	0	0	0	0	0	0	0	0	1288
4	0	0	0	5388	0	0	0	0	1	0	0	5389
5	0	0	0	0	3810	0	0	0	0	0	0	3810
6	0	0	0	0	0	4679	0	0	0	0	0	4679
7	0	0	0	0	0	0	651	0	0	0	0	651
8	0	0	0	0	0	0	0	4255	0	0	0	4255
outlier	32	1	0	0	0	0	0	0	1622	3628	1500	6783
Sum	961	2217	1288	5388	3810	4679	651	4255	1624	3628	1500	30001

The following results are obtained when the external parameter $\rho = 6$.

Table XXIX

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	956	0	0	0	0	0	0	0	0	0	0	956
2	0	2211	0	0	0	0	0	2	0	0	0	2213
3	0	2	1287	0	0	0	0	0	0	0	0	1289
4	0	0	0	5124	0	0	0	0	0	0	1	5125
5	0	0	0	0	3797	0	0	0	0	0	1	3798
6	0	0	0	193	0	4605	0	0	0	0	0	4798
7	0	0	0	0	0	0	649	4253	0	0	0	4902
8	0	0	0	0	0	0	0	0	1608	0	0	1608
9	1	0	0	0	0	0	0	0	0	3590	0	3591
outlier	4	4	1	71	13	74	2	0	16	38	1498	1721
Sum	961	2217	1288	5388	3810	4679	651	4255	1624	3628	1500	30001

9.2 Fuzzy C-means results

The number of clusters in input data sets was 10 and number of input data was 10000 and we increased dimension of data set as follow: 5, 10, 20, 30, 50 and 75. Therefore, we had 6 data sets in our test.

The input clusters and the results with 5-dimensional data sets:

Table XXX

Input Clusters

Clusters	dimensions
1	1,2,3,4
2	1,2
3	1,2,3
4	1,2,3
5	1,2,3,4
6	1,2,3,4,5
7	1,2
8	1,2,3,4
9	1,2,3,4
10	1,2

Table XXXI

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	0	24	0	88	0	0	0	0	581	178	38	909
2	98	53	0	0	0	0	34	0	0	83	53	321
3	63	5	0	123	1472	0	20	0	0	0	80	1763
4	0	33	26	0	0	0	0	0	0	473	47	579
5	11	18	59	28	0	0	3	0	560	110	42	831
6	10	32	22	15	0	0	29	0	702	255	46	1111
7	54	0	49	0	0	0	16	0	0	398	36	553
8	0	50	0	113	0	0	17	0	746	439	100	1465
9	0	0	9	0	0	1191	0	0	0	100	9	1309
10	22	0	23	0	0	0	310	571	0	184	49	1159
Sum	258	215	188	367	1472	1191	429	571	2589	2220	500	10000

The input clusters and the results with 10-dimensional data sets:

Table XXXII

Input Clusters

Clusters	dimensions
1	1,2,3,
2	1,2
3	1,2,3
4	1,2,3
5	1,2,3,4
6	1,2
7	1,2,3
8	1,2,3
9	1,2,3,4
10	1,2

Table XXXIII

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	4	15	0	13	0	55	0	0	8	5	12	112
2	13	26	4	59	0	145	8	0	7	9	13	284
3	5	12	0	13	0	30	4	0	0	1	9	74
4	65	221	91	266	433	704	75	0	49	77	99	2080
5	27	276	4	414	550	964	69	0	56	86	132	2578
6	6	17	0	23	0	71	3	0	3	3	5	131
7	41	71	77	48	0	260	24	7	35	18	29	610
8	187	406	263	88	0	1648	113	665	208	126	192	3896
9	22	25	1	29	0	62	5	0	6	10	4	164
10	16	1	6	4	0	30	2	0	6	1	5	71
Sum	386	1070	446	957	983	3969	303	672	378	336	500	10000

The input clusters and the results with 20-dimensional data sets:

Table XXXIV

Input Clusters

Clusters	dimensions
1	1,2,3,4;5,6,7
2	1,2,3,4,5,6
3	1,2,3,4,5,6,7,8
4	1,2,3,4,5,6,7
5	1,2,3,4
6	1,2,3,4,5
7	1,2,3,4
8	1,2,3,4,5
9	1,2,3,4
10	1,2,3,4

Table XXXV

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	17	0	0	3	1	13	0	37	393	18	10	492
2	0	0	0	0	0	0	0	0	0	4	0	4
3	185	1523	5	213	215	96	0	336	984	122	239	3918
4	0	0	0	0	0	0	0	0	0	1	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0
6	56	0	7	115	22	52	2	26	0	52	47	379
7	6	0	0	1	2	3	0	40	195	2	6	255
8	0	0	0	0	0	0	0	0	0	0	0	0
9	48	0	437	66	7	204	2516	117	489	741	183	4808
10	21	0	17	23	0	16	17	4	0	31	15	144
Sum	333	1523	466	421	247	384	2535	560	2061	971	500	10001

The input clusters and the results with 30-dimensional data sets:

Table XXXVI

Input Clusters

Clusters	dimensions
1	1,2
2	1,2
3	1,2
4	1,2
5	1,2,3
6	1,2
7	1,2
8	1,2
9	1,2
10	1,2,3

Table XXXVII

Contingency table

Input / Output	Clusters										Outlier	Sum
	1	2	3	4	5	6	7	8	9	10		
1	12	361	14	79	56	89	676	51	207	75	85	1705
2	8	11	14	14	23	15	132	4	41	6	10	278
3	39	17	21	77	30	64	408	40	112	7	43	858
4	79	19	41	121	51	75	501	48	85	10	56	1086
5	6	48	18	34	13	60	316	5	86	18	28	632
6	31	5	19	40	43	38	245	10	16	20	24	491
7	6	234	50	89	17	50	445	56	320	11	59	1337
8	64	58	67	76	79	78	593	22	152	32	70	1291
9	43	25	4	45	29	58	389	34	115	19	41	802
10	87	117	32	112	36	74	700	30	220	28	84	1520
Sum	375	895	280	687	377	601	4405	300	1354	226	500	10000

The input clusters and the results with 50-dimensional data sets:

Table XXXVIII

Input Clusters

Clusters	dimensions
1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37
2	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41
3	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38
4	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43
5	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21, 22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37
6	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42
7	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48
8	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45
9	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46
10	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48

Table XXXIX

Contingency table

Input / Output	Clusters										Outlier	Sum
	1	2	3	4	5	6	7	8	9	10		
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	249	1592	1379	446	0	107	469	933	0	714	364	6253
8	0	0	0	0	0	0	0	0	0	0	0	0
9	92	0	0	332	1791	47	0	0	1348	0	136	3746
10	0	0	0	0	0	0	0	0	0	0	0	0
Sum	341	1592	1379	779	1791	154	469	933	1348	714	500	10000

The input clusters and the results with 75-dimensional data sets:

Table XL

Input Clusters

Clusters	Dimensions
1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50, 51,52,53,54,55,56,57
2	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50, 51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74
3	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50, 51,52,53,54,55,56,57,58
4	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50, 51,52,53,54,55,56,57,58,59,60
5	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50, 51,52,53,54,55,56,57,58
6	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42, 43,44,45,46,47,48
7	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50, 51,52,53,54,55,56,57
8	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29, 30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54
9	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30, 31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57
10	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,

Table XLI

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0
5	24	0	415	693	0	225	270	0	0	1797	196	3620
6	0	0	0	0	0	0	0	0	0	0	0	0
7	972	0	0	0	720	210	143	1201	698	0	304	4248
8	0	2131	0	0	0	0	0	0	0	0	0	2131
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0	0	1
Sum	996	2131	415	693	720	435	415	1201	698	1797	500	10001

9.3 K-means results

We didn't experimented k-means clustering methods on 300-dimensional data sets because only 5 dimensions is enough to show that k-means isn't suitable for clustering data with subspace clusters. The input clusters and results with 5-dimensional data sets where each cluster have different subspaces are as follow:

Table XLII

Input Clusters

Clusters	dimensions	Points
1	1,2,3,4	258
2	1,2	215
3	1,2,3	188
4	1,2,3	367
5	1,2,3,4	1472
6	1,2,3,4,5	1191
7	1,2	429
8	1,2,3,4	571
9	1,2,3,4	2589
10	1,2	2220

Table XLIII

Contingency table

Input / Output	Clusters 1	2	3	4	5	6	7	8	9	10	Outlier	Sum
1	77	2	0	29	725	0	22	0	0	0	35	890
2	0	7	0	42	747	0	0	0	0	0	44	840
3	0	48	24	0	0	0	0	0	0	643	48	763
4	0	30	80	0	0	0	0	0	0	595	58	763
5	0	19	0	0	0	0	0	0	0	518	39	576
6	0	7	0	296	0	0	64	0	0	0	111	478
7	0	59	0	0	0	0	0	0	1366	0	52	1477
8	0	2	0	0	0	0	122	295	0	273	48	740
9	181	24	29	0	0	0	221	276	0	0	33	764
10	0	17	55	0	0	1191	0	0	1223	191	32	2709
	0	0	0	0	0	0	0	0	0	0	0	0
Sum	258	215	188	367	1472	1191	429	571	2589	2220	500	10000

The input clusters and the results with 5-dimensional data sets where each clusters have the same subspaces are as follow:

Table XLIV

Input Clusters

Clusters	dimensions	Points
1	1,2,3,4,5	956
2	1,2,3,4,5	476
3	1,2,3,4,5	740
4	1,2,3,4,5	1613
5	1,2,3,4,5	376
6	1,2,3,4,5	926
7	1,2,3,4,5	447
8	1,2,3,4,5	987
9	1,2,3,4,5	1242
10	1,2,3,4,5	1736

Table XLV

Contingency table

Input / Output	Clusters										Outlier	Sum
	1	2	3	4	5	6	7	8	9	10		
1	0	0	0	1607	0	0	0	0	0	3	20	1630
2	0	0	0	0	0	0	0	0	0	0	182	182
3	0	476	0	0	0	0	0	0	0	0	39	515
4	0	0	740	0	376	0	0	0	0	0	48	1164
5	956	0	0	0	0	0	0	0	0	0	35	991
6	0	0	0	0	0	0	0	0	1242	0	95	1337
7	0	0	0	0	0	0	0	987	0	0	25	1012
8	0	0	0	0	0	926	0	0	0	0	27	953
9	0	0	0	6	0	0	0	0	0	1733	11	1750
10	0	0	0	0	0	0	447	0	0	0	18	465
	0	0	0	0	0	0	0	0	0	0	0	0
Sum	956	476	740	1613	376	926	447	987	1242	1736	500	9999

REFERENCES

[1] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, 1996.

[2] Bradley, P.S. *Data Mining as an Automated Service*, p.1-13 Advances in Knowledge Discovery and Data Mining: 7th Pacific-Asia Conference, PAKDD 2003. Proceedings, Editors: K.-Y. Whang, J. Jeon, K. Shim, J. Srivastava (Eds.), Springer-Verlag Heidelberg, Seoul, Korea, April 30 - May 2, 2003

[3] Kantardzic Mehmed, *Data Mining: Concepts, Models, Methods, and Algorithms*, John Wiley & Sons, IEEE Press 2003

[4] Becher Jonathan D., Berkhin Pavel, Edmund Freeman, *Automating Exploratory Data Analysis for Efficient Data Mining*, KDD 2000, ACM, Boston MA, 2000

[5] Schurmann, J. *Pattern Classification. A unified view of statistical and neural approaches*, John Wiley & Sons, New York, NY 1996

[6] Zhengxin, Chen., *Data Mining and Uncertain Reasoning: An integrated approach*, New York, John Wiley& Sons, 2001

[7] Agrawal, R. Srikant., *Fast algorithms for mining association rules in large databases*, Proc International Conference on Very Large Databases, pp. 478-499. Santiage, Chile: Morgan Kaufmann, Los Altos, CA, 1994

[8] Witten, Ian H., Frank, Eibe. *Data Mining: Practical machine learning tools with Java implementations*, San Francisco, Morgan Kaufmann, 2000

[9] Padhraic Smyth. *Clustering using Monte Carlo cross-validation*, in Knowledge Discovery and Data Mining, pages 126-133, 1996

[10] Object Management Group (OMG), *Common Warehouse Metamodel (CWM) Specification*, version 1.1, p.12-1, 12-114, http://www.omg.org/cwm/

[11] Jürgen Lind, MASSIVE: Software Engineering for Multiagent Systems, German Research Center for AI (DFKI), Saarbrücken, Germany, 1999

[12] Werner Eric, *Cooperating agents: a unified theory of communication and social structure*, Morgan Kaufmann Series In Research Notes In Artificial Intelligence: Distributed artificial intelligence: vol. 2 archive, San Francisco, CA, US, 1990

[13] Klusch, Matthias., Lodi, Stefano., Moro, Gianluca, *The role of agent in Distributed Data Mining: Issues and Benefits*, Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology, IEEE Press 2003

[14] Java Community Process, *Java Specification Request 73: Java Data Mining (JDM)*, version final, http://www.jcp.org/en/jsr/detail?id=73

[15] Yuxiao Li, George Benwell, Peter Whigham, Nick Mulgan, *An Agent-oriented software engineering paradigm and the design of a new generation of spatial information system*, SIRC 2000 – The 12th Annual Colloquium of the Spatial Information Research Centre, University of Otago, Dunedin, New Zealand, December 10-13th 2000

[16] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley, 2002

[17] Kargupta, H., Hamzaoglu, I., Stafford, B. *Scalable, Distributed Data Mining Using An Agent Based Architecture*, Proceedings of Knowledge Discovery And Data Mining. Eds: D. Heckerman, H. Mannila, D. Pregibon and R. Uthurusamy. p. 211-214, AAAI Press, 1997

[18] H. Kargupta, B. Park, D. Hershberger and E. Johnson, *Collective data mining: a new perspective toward distributed data mining*, In H. Kargupta and P. Chan (eds.) Advances in Distributed and Parallel Knowledge Discovery, AAAI Press 1999.

[19] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky. *Papyrus: a system for data mining over local and wide area clusters and super-clusters*. In Proc. Conference on Supercomputing, page 63. ACM Press, 1999

[20] Salvatore Stolfo, Andreas L. Prodromidis, Shelley Tselepis, Wenke Lee, Wei Fan, *JAM: Java agents for Meta-Learning over Distributed Databases*, in Proc. KDD-97, PAGES 74-81, Newport Beach, California, USA, 1997.

[21] Brian Henderson-Sellers and Paolo Giorgini, *Agent-Oriented Methodologies*, Idea Group Publishing, 2005

[22] Joseph P. Bigus, Jennifer Bigus, *Constructing Intelligent Agents with Java: A Programmer's Guide to Smarter Applications*, John Wiley & Sons, 1997

[23] KIF web site, http://logic.stanford.edu/kif/kif.html

[24] PMML web site, http://sourceforge.net/projects/pmml

[25] RDF web site, http://www.w3.org/RDF/

[26] Core J2EE Patterns – Data Access Object, http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

[27] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, *When is "nearest neighbors" meaningful?*, in Proc. 7th Int. Conf. Database Theory, 1999, pp. 217-235

[28] S. Goil, H. Nagesh, and A. Choudhary. *Mafia: Efficient and scalable subspace clustering for very large data sets*, Technical Report CPDC-TR-9906-010, Northwestern University, 2145 Sheridan Road, Evanston IL 60208, June 1999

[29] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[30] Yongqiang Cao and Jianhong Wu, *Dynamics of Projective Adaptive Resonance Theory Model: The Foundation of PART Algorithm*, IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 15, NO. 2, MARCH 2004

[31] Duda, Richard O., Hart, Peter E., Hart, David G. Stork, *Pattern Classification, Second Edition*, Wiley, 2000

[32] L.Breiman, J.H., R.A. Olshen, C.J. Stone. *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1984.

[33] S. Cost and S. Salzberg. *A weighted nearest neighbor algorithm for learning with symbolic features*. Machine Learning, 10:57–78, 1993

[34] FIPA-OS toolkit, *http://fipa-os.sourceforge.net*,

[35] Quinlan J.R. *C4.5:Programs For Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993

[36] FIPA web site, *http://fipa.org*

[37] Padhraic Smyth. *Clustering using Monte Carlo cross-validation*, in Knowledge Discovery and Data Mining, pages 126-133, 1996

[38] Steven P. Fonseca, Martin L. Griss, Reed Letsinger, *Agent Behavior Architectures A MAS Framework Comparison*, AAMAS'02, ACM, Bologne, Italy, 2002

[39] Brian Henderson-Sellers and Paolo Giorgini, *Agent-Oriented Methodologies*, Idea Group Publishing, 2005

[40] PASSI web site, http://mozart.csai.unipa.it/passi/

[41] Publicly Available Agent Platform Implementations, http://www.fipa.org/resources/livesystems.html

[42] Core J2EE Patterns – Data Access Object, http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

[43] W3C. (1999). Resource Description Framework. (RDF), Model and Syntax Specification. *W3C Recommendation 22-02-1999*. Available online *http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/*

[44] Searle, J.R., Speech Acts. Cambridge University Press, 1969.

[45] Communicative Act Library Specification. *FIPA Document #FIPA00037*. Available online *http://www.fipa.org/specs/fipa00037/*

[46] Resource Description Framework (RDF) Schema Specification 1.0, World Wide Web Consortium, 2004. Available online http://www.w3.org/TR/rdf-schema/

[47] JADE Online documentation, http://jade.tilab.com/doc/

[48] Jade Tutorial and Primer, http://www.iro.umontreal.ca/~vaucher/Agents/Jade/Ontologies.htm

[49] Kenneth J. McGarry, Stefan Wermter and John MacIntyre, *Knowledge Extraction from Radial Basis Function Networks and Multi-layer Perceptrons*, University of Sunderland, IEEE, Sunderland, England, 1999

[50] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park, *Fast algorithms for projected clustering*, in Proc. SIGMOD'99, 1999, pp.61-72

[51] Lance Parsons, Ehtesham Haque, Huan Liu, *Subspace Clustering for High Dimensional Data: A Review*, SIGKDD Exploration, Volume 6, Issue 1 pp. 90-105, 2004