3. Introduction.

The vast amount of heterogeneous information source available on the Internet, demands advanced solutions for acquiring, mediating and maintaining relevant information for the common user. In most organizations two factors play a crucial role: managing the knowledge that is necessary for doing business and managing the hardware and software infrastructure that supports the business processes. Usually, business processes and infrastructure are not optimally aligned. To complicate matters even further, the software assets are continually evolving over time due to changed business insights and technical evolution of the platform on which the software depends. Therefore, the investigation of how technologies can be applied to enable the creation, consolidation, conservation and continuous utilization of knowledge about valuable software and hardware systems that are part of infrastructure, has become necessary. The Computation is taking different form due to the fast growth in computer technology. This will definitely make every thing, more intelligent than found today, resulting into the virtual world. Though we are heading towards this dream, but still many issues need to be addressed in their completeness to be applied in totality for this virtual world.

3.1. History of Network Computing:

Mobile-agents can revolutionalize the design and development of distributed systems. To put this claim into perspective, a brief overview of three programming paradigms for distributed computing are discussed here-

- Client-server
- Code-on-demand
- Mobile agents

Client-Server Paradigm:
In this paradigm, a server advertises a set of services that provide access to some resources. The server hosts the code that implements these services locally. We say that the server holds the ‘know-how’. Finally, it is the server itself that executes the service
and thus has the processing capability. If the client is interested in accessing service hosted by the server, the client will simply use one or more of the services provided by the server. Note that the client needs some intelligence to decide which of the services it should use. The server has all the 'know-how', resources, and processing power. So far, most distributed systems have been based on this paradigm. It is supported by a wide range of technologies such as remote procedure calling, object request brokers (CORBA), and Java RMI.

![Client-server paradigm](image1)

**Figure-3.1 : Client-server paradigm**

**Code-On-Demand Paradigm:**
According to this paradigm, we first get the 'know-how' when we need it. Suppose, that client initially, is unable to execute its tasks because of lack of code, a host in the network provides the needed code. Once the client receives the code, the computation is carried out in the client. The client holds the processing capability as well as the local resources. Java applets and servlets are the excellent practical examples of this paradigm.

![Code-on-demand paradigm](image2)

**Figure-3.2 : Code-on-demand paradigm**
Mobile Agent Paradigm:

A key characteristic of the mobile-agent paradigm is that any host in the network is allowed a high degree of flexibility to possess any mixture of 'know-how', resources and processors. Its processing capabilities can be combined with local resources. Know-how (in the form of mobile-agents) is not tied to a single host but rather available throughout the network.

3.2. What is Mobile-Agent?

An agent is a software program that assists people and acts on their behalf. They function by allowing people to delegate work to them. They have the ability to interact with their execution environment and to act asynchronously and autonomously upon it. An agent is a software object that is situated within an execution-environment. It has the following characteristics:

- Reactive: It senses changes occurring in the environment and acts accordingly.
• Autonomous: It has control over its own actions.
• Goal-Driven: It is pro-active.
• Temporally continuous: It continually executing.

In addition to the aforesaid properties, it may have the following orthogonal properties as well:

• Communicative: able to communicate with other agents.
• Mobile: It can travel from one host to another.
• Learning: Adapts in accordance with previous experience.

Mobility is an orthogonal property of agents, and all agents are not mobile. An agent can just sit at one host and communicate with its surroundings by conventional means, such as various forms of remote procedure calls and messages. Agents that cannot move are called stationary agents. The stationary agents executes only in the system where it begins execution. If it needs information which is not available in the system or needs to interact with an agent on a different system, it typically uses a communication mechanism such as remote procedure call (RPC). In contrast mobile-gents are not bound to the system where they begin execution. They have a unique ability to transport itself from one system in the network to another system with their execution states. The ability to travel allows a mobile-gent to move to a system that contains an object with which the agent wants to interact and then take advantage of being in the same host or network as the object. Mobile-agents create new paradigm for data-exchange and resource-sharing in rapidly growing and continually changing computer networks.

Why Mobile Agents?

Mobile agents are not employed as a consequence of technology available, but by the advantages they provide for the creation of distributed systems. Here are the seven good reasons for using mobile agents:

1. They reduce network load.
2. They overcome network latency.
3. They encapsulate protocols.
4. They execute asynchronously and autonomously.
5. They adapt dynamically.
6. They are naturally heterogeneous.
7. They are robust and fault-tolerant.

They reduce the network load:

Distributed systems often rely on communications protocols that involve multiple interactions to accomplish a given task, resulting in a lot of network traffic. Mobile-agents allow us to package a conversation and dispatch to a destination host, where the interactions take place locally. The motto is simple: “Move the computations to the data rather than the data to the computation”.

![Figure-3.4: RPC based approach](image1)

![Figure-3.5: Mobile-Agent based approach](image2)

They overcome network latency:

Critical real-time systems, such as robots in manufacturing processes, need to respond in real time to changes in their environments. Controlling such systems through a factory network of a substantial size involves significant latencies, which is not acceptable for critical systems. Mobile agents offer a solution, because they can be dispatched from a central controller to act locally and execute the controller’s directions.
They encapsulate protocols:

In a distributed system, each host implements the protocols needed to code the outgoing and incoming data. It often is a cumbersome task to upgrade protocol properly. As a result, protocols often become a legacy problem. But Mobile-agents can move to remote hosts to establish “channels” based on propriety protocols.

They execute asynchronously and autonomously:

Often mobile devices use the expensive network connections. Tasks that require a continuous open connection between a mobile device and a fixed network probably will not be economically and technically feasible. Thus tasks may be embedded into mobile agents, which can then be dispatched into the network. Thus mobile agents act independently and asynchronously, independent of the creating process.

They adapt dynamically:

Mobile-agents can sense their execution environment and react autonomously to changes. Multiple mobile-agents can distribute themselves among hosts in a network so as to maintain an optimal configuration for solving a particular problem.

They are naturally heterogeneous:

Network computing is fundamentally heterogeneous, often from both hardware and software perspective. Because mobile-agents are generally computer and transport-layer independent and are dependent only on their execution environment, they provide optimal conditions for seamless system integration.

They are robust and fault-tolerant:

The ability of the mobile-agents to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems. Suppose, if a host is being shut down, all agents executing there will be warned and given time to dispatch and continue their operation on another host in the network.
3.2.1. Applications of Mobile-Agents:

There are at least three ways to view the role of mobile-agent technology:

- As a communication mechanism
- As a data transport vehicle among hosts
- As a framework for partitioning application functionality

The various applications that benefit from the mobile agent paradigm are as follows:

- **Electronic commerce:**
  Mobile agents are well suited for e-commerce. A commercial transaction may require real-time access to remote resources such as stock quotes and perhaps agent-to-agent negotiation. Different agents will have different goals and will implement and exercise different strategies to accomplish these goals.

- **Personal assistance:**
  The mobile-agent's ability to execute on remote hosts makes it suitable for playing a role as an assistant capable of performing tasks in the network on behalf of its creator. The remote assistant will operate independently of its limited network connectivity, and the creator can feel free to turn off his or her computer.

- **Secure brokering:**
  An interesting application of mobile-agents is in collaboration in which not all the collaborators are trusted. In this case, the parties could let their mobile agents meet on a mutually agreed secure host, where collaboration can take place without the risk of the host taking the side of one of the visiting agents.

- **Distributed information retrieval:**
  Information retrieval is a popular example of a mobile-agent application. Instead of moving large amounts of data to the search-engine so that it can create search
indexes, you dispatch agents to remote information sources, where they locally create search indexes that can later be shipped back to the origin.

- **Telecommunication networks services:**

  Support and management of advanced telecommunication services are characterized by dynamic network reconfiguration and user customization. The physical size of these networks and the strict requirements under which they operate call for mobile agent technology to form the glue that keeps such systems flexible, yet effective.

- **Workflow applications and groupware:**

  It is in the nature of workflow applications to support the flow of information between co-workers. The mobile agent is particularly useful here, because in addition to mobility, it provides a degree of autonomy to the workflow item.

- **Monitoring and notification:**

  This classic mobile-agent application highlights the asynchronous nature of mobile-agents. An agent can monitor a given information source without being dependent on the location from which it originates.

- **Information dissemination:**

  Mobile-agents embody the so-called Internet push model. Agents can disseminate information such as news and automatic software updates for vendors.

- **Parallel processing:**

  Given that, mobile-agents can create clones in the network, one potential use of the mobile-agent technology is to administer parallel processing tasks. If a computation requires so much processor power that it must be distributed among multiple processors, an infrastructure of mobile agent hosts could be a possible way to allocate the processes.
3.2.2. Features of Mobile-Agents:

The two fundamental elements of the system are an agent, and the environment in which they execute, which is termed as the place. An agent has five attributes namely the state, which is required for the agent to resume computation after traveling, the implementation, which is required for location independent agent execution, the interface which is needed for agent communication, the identifier, which is needed to recognize and locate traveling agents and finally the principals that are needed to determine legal and moral responsibility. Each attributes are discussed in detail below:

State:
When an agent travels, it transports its state with it. It must do so in order to resume execution at the destination-host, a characteristic of virtually all the mobile-agents. The agent’s state at any given time is a snapshot of its execution. For most programming languages, we can partition the agent’s state into its execution state, which is its runtime state (including its program counter and frame stack), and its object state, which is the value of the instance variables in the object.

Implementation:
Like any other computer program, a mobile-agent needs a code to execute. When it travels, it has the option of either taking its implementation code or going to the destination, seeing what code is already there, and retrieving any missing code over the network (code-on-demand). The agent implementation should be both executable by the destination host and safe for the host to execute.

Interface:
An agent provides an interface that allows other agents and systems to interact with it. This interface can be anything from a set of method signatures that allow other agents and applications to access methods on the agent to a messaging interface that allows agents to communicate.
**Identifier:**
Every agent has an identifier that is unique during its lifetime (immutable). Agents require identities so that they may be identified and located via, for example, directory services. Because an agent's identifier is globally unique and immutable it can be used as a key in operations that require a way of referring to a particular agent instance.

**Principals:**
A principal is an entity whose identity can be authenticated by any system that the principal may try to access. A principal can be an individual, an organization or a corporation. An identity consists of a name and possibly other attributes. For agents, we identify at least two main principals namely *manufacturer*, who is the author, and the *owner*, who is the principal having the legal and moral responsibility for the agent's behavior.

Agents travel between places. The most common view of a place is that it is a context in which an agent can execute. It may be regarded as the entry point for a visiting agent that wishes to execute. The place provides a uniform set of services that the agent can rely on irrespective of its location.

![Figure-3.6 : Place and Engine](image)

Four concepts play an important role in the environment of an agent execution. They are:
• **Engine**: This is considered to be the workhorse and virtual machine for one or more places. It provides places and agents with links to underlying network and other resources provided by the host.

• **Resources**: Together, the engine and the place provide controlled access to local resources and services such as: networks, databases, processors and memory disks, and other hardware and software services.

• **Location**: This is an important concept for mobile-agents. It is defined as the location of an executing agent as the combination of the name of the place in which it executes and the network address of the engine in which the place resides.

• **Principals**: Like an agent, a place has two principals. A place is associated with the authorities that identify the person or organization for which the place acts (place master), as well as the manufacturer of the place. The manufacturer is the provider of the place implementation, and the place master is the principal that has the responsibility for the operation of the place.

**Agent Behavior: Creation and Disposal**:

An agent gets created in a place. The creation can be initiated by another agent residing in the same place or by another agent or non-agent system outside the place. The creator is required to authenticate itself to the place, establishing the authority and credentials that the new agent will possess. The creator can also supply initialization arguments for the agent. The class definition needed to instantiate the agent can be present on the local host or a remote host. Creation involves three steps:

1. **Instantiation and identifier assignment**: The class definition is loaded and made executable, and the agent-object is instantiated. The agent class specifies both the interface and the implementation of the agent. The place assigns a unique identifier to the agent.

2. **Initialization**: The agent is given a chance to initialize itself using any initialization arguments provided by the creator. Only when the initialization has been completed, agent assumes that it has been fully and correctly installed in its place.
3 **Autonomous Execution:** After being fully installed in the place, the agent starts execution. It is now capable of executing independently of other agents in the same place.

The disposal of an agent can be initiated by the agent itself, by another agent residing in the same place, or by another agent or non-agent system outside the place. An agent can also be disposed of by the system for one of the following reasons:

- **End of Lifetime**: The given lifetime of the agent has expired.
- **No use**: No one refers to or uses the agent.
- **Security violation**: The agent has violated security rules.
- **Shutdown**: The system is shutting down.

Disposing of an agent is a two-step process:

1. **Preparing for disposal**: The agent is given a chance to finalize its current task before it is disposed of.
2. **Suspension of Execution**: The place suspends the execution of the agent.

**Agent Behavior: Transfer**

The transfer process can be initiated by the agent itself, by another agent residing in the same place, or by another agent or non-agent system outside the place. The agent is dispatched from its current place (origin) and received by the specified place (destination). The dispatch process is managed by the origin and destination places. When the origin place contacts the destination place, the destination place either can fulfill the travel request or return a failure indication to the origin. If the origin place cannot contact the destination place, it must return a failure indication to the agent.
**Dispatching an Agent:**

When a mobile-agent is preparing for a trip, it must be able to identify its destination; otherwise it runs in a default place selected by the destination agent system. Once the location of the destination is established, the mobile-agent informs the local agent system that it wants to transfer itself to the destination agent system.

This message is relayed via an internal API between the agent and the agent system. When the agent system receives the agent’s trip request, it should do the following:

1. **Suspend the agent:** The agent is warned about the imminent transfer and is allowed to prepare for departure (complete its current task). When that is done, the execution thread is halted.

2. **Serialize the agent:** The agent-state and agent-class are serialized by the engine. Serialization is the process of creating a persistent representation of the agent object that can be transported over a network. Serialization of the agent may include the execution state.
3. **Encode the serialized agent**: The engine now encodes the serialized agent for the chosen transport protocol.

4. **Transfer the agent**: The engine establishes a network connection to the specified destination host and transfers the encoded serialized agent.

**Receiving an Agent:**

Before an engine receives an agent, it must determine if it can accept an agent from the sending host. Only after the sender has successfully authenticated itself to the receiving engine, the actual data-transfer take place. The steps involved are:

1. **Receive the agent**: When the destination agent agrees to the transfer, the encoded agent is received.

2. **Decode the agent**: The engine decodes the incoming data stream.

3. **Deserialize the agent**: The persistent representation of the agent is deserialized. The agent class is instantiated, and the transferred agent state is restored.

4. **Resume agent execution**: The recreated agent is notified of its arrival at the destination place. It can now prepare to resume its execution and is given a new thread of execution.
Aglet Class Transfer:
The agent cannot resume execution in the destination-engine without its class being present. There are several ways to make the class available for the destination-engine, depending on the location of the class:

- **Class at destination:** If the class is already available at the destination, either in the engine’s class cache or in the local file system, there is no need to transfer the class. The transferred agent need contain only the information required to identify the class, such as full class name and discriminator. It may also contain additional information that describes the location of the class definition.

- **Class at the origin:** If the class is located at the origin, it can easily be transported with the agent’s state to the destination-engine.

- **Code on demand:** The class is available from the server and the destination-engine can retrieve the class on a code-on-demand basis.

Figure-3.8 : Agent class transfer at destination, origin and server
After an agent has been instantiated, it often creates other objects. These are needed for their instantiation and continued execution. If any of these classes are not available at the destination-engine, it must be transferred from either from the origin or the sender of the agent. Either the agent can decide to bring all the needed classes in the first place, or auxiliary classes can be brought on a per-request basis.

3.3. Pervasive Computing:

Ubiquitous/Pervasive object can be defined as an intelligent, autonomous computing object, which can be accessed by anything at any time and anywhere. Mark Weiser put the vision of Ubiquitous computing forward at the beginning of the last decade in his influential article “The Computer for the 21st Century”. Weiser foresaw omnipresent computers that serve people in their everyday lives at home and work, functioning invisibly and unremarkably in the background, freeing the system from tedious routine tasks. This vision, however, remained only a subject for few researchers until the end of that decade. This has changed dramatically over the last few years [26].

The advances in computing and communication technologies and software have resulted in an explosive growth in computing systems and applications that impact all aspects of our life. However, as the scale and complexity of these systems and applications grow, their development, configuration and management-challenges are beginning to break current paradigms, overwhelm the capabilities of existing tools and methodologies, and rapidly render the systems and applications brittle, unmanageable and insecure [22]. In mid October, 2001, IBM released the manifesto observing that the main obstacle to further progress in the IT industry is a threatening software complexity crisis [17]. The manifesto pointed out the difficulty of managing computing systems [18]. The need to integrate several heterogeneous environments into corporate wide computing systems and to extend that beyond company boundaries into the Internet, introduces new level of complexities. This has led researchers and industrialists to consider ubiquitous computing to successfully deal with similar challenges of complexity, dynamism, heterogeneity and uncertainty. Dreams of a more convenient, healthier, safer and happier life through the advent of ubiquitous, Information Technology (IT) has been clearly expressed [27]. Already the macro systems that surround us, for example, intelligent buildings, electronic
cars and hosts of portable devices have brought about the dynamic and prominent changes in our daily lives. Since Mark Weiser [28] wrote his influential article, technology has advanced along many dimensions. In addition to the well-known developments in capabilities such as processing power and storage in portable devices, other significant developments have occurred. These developments are likely to affect our ability to deploy ubiquitous computing systems. They include new technologies such as Global Positioning Systems (GPS), Smart Cards and Radio Frequency Identification (RFID) tags and social development such as the increasingly widespread acceptance of video surveillance in public places. The emergence of the web as global information and service resource and the widespread adoption of digital mobile telephony, letting users experience nearly ubiquitous wireless communications.

**Structure of Ubiquitous Object:**
Ubiquitous object can be defined as an intelligent, autonomous computing object, which can be accessed by anything at any time and anywhere. The structure of ubiquitous object is as shown in the Figure-3.9. Every ubiquitous object is identified by its unique object Id (Ex: RFID tag). Each such ubiquitous object will be responsible for managing its own internal state, behaviour and managing its interactions with other ubicomp compliant objects. One object after identifying the other ubicomp compliant object near by, stores its identity in the knowledge-base. Depending on the type of the object and other relevant information the RFID tag indicates, the execution unit will initiate the necessary actions.
Ubicomp applications should have the following characteristics [31]:

- Ubiquitous Access
- Context Awareness
- Intelligence
- Natural Interaction

**Ubiquitous Access** refers to a situation in which users are surrounded by a multitude of sensors. The sensors enable the implicit input from a "physical world" into a "virtual world" will be operated in a time driven or event driven way and actuators as the generic means for implicit output from the "virtual" to the physical world "will respond to the surrounding in either a reactive or proactive fashion.

**Context awareness** refers to the ability of the systems to recognize and localize objects as well as people and their intentions. The context of an application can be understood as "any information that can be used to characterize the situation of an entity", an entity being a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves. A key architecture design principle for context-aware applications will be to decouple mechanisms for collecting or sensing context information and its interpretation. This provision and
exploitation of the information helps in building context-aware applications. To support building the context-aware applications, software developers should not be concerned with how, when and where context information is sensed. Sensing context must happen in an application independent way and context representation must be generic for all possible applications.

**Intelligence** refers to the fact that a technology-rich environment is able to adapt itself to the people that live in it, learn from their behavior and possibly recognize and as well as show emotions.

**Natural Interaction** refers to advanced modalities like natural speech and gesture recognition as well as speech synthesis, which will allow a much more human-like communication with the digital environment that is possible today.

### 3.3.1. Computational Trends:

The Computation is taking a different form due to the fast growth in computer technology. This will surely make every object, more intelligent than found today, resulting into the virtual world of ubicomp world. Though we are heading towards this dream, but still many issues need to be addressed in its completeness to be applied in totality for this virtual world.

The figure-3.10 illustrates the major trends in computing [29].

![The Major Trends in Computing](image-url)

Figure- 3.10 :Trends in computing.
3.3.2. Applications:

Ubiquitous Applications:

The paper (1991) by Mark Weiser that describes the vision of ubiquitous computing highlights the merging of profound technologies in to everyday life [30]. Today, ubicomp applications are diverse in nature ranging from small applications that help commuters track train and bus schedules, to smart laboratories, smart museums, instrumented classrooms, etc. The ubiquitous computing environment may contain many devices with which user interact. Speech, gestures and even a physical interaction with the devices can be used as interaction modalities. Behaviour by the users may cause actions in the physical world. For example, lying down in an intelligent room can cause the window curtains to close, the lights to dim, and the music to be turned off. Both input and output in ubicomp environment may be distributed. Additionally as ubicomp occurs everywhere, there may be a large number of users interacting with a system simultaneously.

3.4. Data Mining (DM):

Data mining is the process of automatically extracting new and useful knowledge hidden in the large data-sets. Data mining is a multidisciplinary field, drawing work from areas including database technology, artificial intelligence, machine learning, neural networks, statistics, pattern recognition, knowledge-based systems, knowledge acquisition, information retrieval, high performance computing, and data visualization. This emerging discipline is becoming increasingly important as advances in data collection lead to the explosive growth in the amount of available data. Data mining techniques primarily help to analyze the commercial data sets and play a critical role in analyzing and understanding purchasing behaviors for effective consumer relations management, process optimization, personalized marketing, and customer segmentation. Data mining’s success has sparked an interest in applying such analysis techniques to various scientific and engineering fields, such as biology, medicine, astronomy, ecosystems modelling, and structural mechanics. Most organizations can be currently labeled as ‘Data rich’, since they are collecting increasing volumes of data about business process and resources. The
explosive growth of many business, government and scientific databases has far outpaced our ability to interpret and digest this data. We are drowning in information yet starving for knowledge. Data-mining therefore appears as a useful tool to address the need for extracting useful information such as hidden pattern from the databases. The application of DM is illustrated in the example given below:

**Example:** The information that customers who purchase a keyboard (K) also tend to buy a mouse (M) at the same time is represented as pattern, called 'association rule' and is represented as:

\[ K \Rightarrow M \text{ [support = 60\%, confidence = 70\%].} \]

Given a set of items \( I = \{ I_1, I_2, I_3, \ldots, I_m \} \) and the database \( D \), with transactions = \( \{ t_1, t_2, t_3, \ldots, t_n \} \) where \( t_i = \{ I_{i1}, I_{i2}, \ldots, I_{ik} \} \) and \( I_{ij} \in I \), an association rule is the implication of the \( X \Rightarrow Y \) where \( X, Y \subseteq I \) are sets of items called as item sets and \( X \cap Y = 0 \). A set of items is referred to as 'item set'. An item set satisfies minimum support i.e if the occurrence frequency of item set is greater than or equal to minimum support, then it is a 'frequent item set'. Association rule mining is to find all such frequent item sets. Based on the types of values, the association rules can be classified into two categories: Boolean Association Rules and Quantitative Association Rules.

**Boolean Association Rule**: \( K \Rightarrow M \text{ [support = 60\%, confidence = 70\%].} \)

**Quantitative Association Rule**: \( (\text{Age} = 26..30) \Rightarrow (\text{cars} = 1..2) \text{ [support = 20\%, confidence = 45\%].} \)

Support of an association pattern refers to the percentage of task relevant data transactions for which the pattern is true, i.e:

\[
\text{Support (} A \Rightarrow B \text{)} = \frac{\text{No. of tuples containing both A & B}}{\text{Total No. of tuples}}
\]
Confidence is defined as the measure of certainty or trustworthiness associated with each discovered pattern.

\[
\text{Confidence} (A \rightarrow B) = \frac{\text{No. of tuples containing both A & B}}{\text{No. of tuples containing A}}
\]

Figure 3.11 illustrates frequent itemsets and support value use.

<table>
<thead>
<tr>
<th>Transaction number</th>
<th>Items</th>
<th>Support</th>
<th>Frequent Item sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B C D</td>
<td>5</td>
<td>B, F</td>
</tr>
<tr>
<td>2</td>
<td>A B E F</td>
<td>4</td>
<td>A, AB, AF, ABF, BF, C, E, EF</td>
</tr>
<tr>
<td>3</td>
<td>A B E F</td>
<td>3</td>
<td>AE, ABE, ABEF, AEF, BC, BE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BEF, CF</td>
</tr>
<tr>
<td>4</td>
<td>A B C F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>A B C E F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>C D E F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.11: An example of a dataset along with its frequent itemsets.

As an illustration consider the database as shown in figure 3.12.

<table>
<thead>
<tr>
<th>TID</th>
<th>List of item IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>I1, I2, I5</td>
</tr>
<tr>
<td>T2</td>
<td>I2, I4</td>
</tr>
<tr>
<td>T3</td>
<td>I2, I3</td>
</tr>
<tr>
<td>T4</td>
<td>I1, I2, I4</td>
</tr>
<tr>
<td>T5</td>
<td>I1, I3</td>
</tr>
<tr>
<td>T6</td>
<td>I2, I3</td>
</tr>
<tr>
<td>T7</td>
<td>I1, I3</td>
</tr>
<tr>
<td>T8</td>
<td>I1, I2, I3, I5</td>
</tr>
<tr>
<td>T9</td>
<td>I1, I2, I3</td>
</tr>
</tbody>
</table>

Figure-3.12: Data set for Association Rule Mining.
Here $T_1 \ldots T_9$ refers to the transactions and $I_1 \ldots I_n$ refers Items purchased in each transaction $T_i$.

<table>
<thead>
<tr>
<th>1-Itemsets</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

**Figure-3.13:** 1-itemsets for Association Rule Mining.

The various process of finding $k$-itemsets is shown below:

**PRUNE:** Count $<$ Support then eliminate them.

**JOIN**

### 1-Itemsets Support Count

<table>
<thead>
<tr>
<th>1</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

### Minimum Support $= 2$

### 2-Itemsets Support Count

<table>
<thead>
<tr>
<th>1,2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3</td>
<td>4</td>
</tr>
<tr>
<td>1,4</td>
<td>1</td>
</tr>
<tr>
<td>1,5</td>
<td>2</td>
</tr>
<tr>
<td>2,3</td>
<td>4</td>
</tr>
<tr>
<td>2,4</td>
<td>2</td>
</tr>
<tr>
<td>2,5</td>
<td>2</td>
</tr>
<tr>
<td>3,4</td>
<td>0</td>
</tr>
<tr>
<td>3,5</td>
<td>1</td>
</tr>
<tr>
<td>4,5</td>
<td>0</td>
</tr>
</tbody>
</table>

### 3-Itemsets Support Count

| 1,2,3 | 2 |
| 1,2,5 | 2 |

### 2-Itemsets Support Count

<table>
<thead>
<tr>
<th>1,2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3</td>
<td>4</td>
</tr>
<tr>
<td>1,5</td>
<td>2</td>
</tr>
<tr>
<td>2,3</td>
<td>4</td>
</tr>
<tr>
<td>2,4</td>
<td>2</td>
</tr>
<tr>
<td>2,5</td>
<td>2</td>
</tr>
</tbody>
</table>
The major issues in data mining are: mining methodology, user interaction, performance, and diverse data types. Local and wide area networks (such as the Internet) connect many sources of data, forming huge, distributed, and heterogeneous databases. The discovery of knowledge from different sources poses great challenges to data-mining. Few challenges have been addressed in recent data mining research and development, to a certain extent, and are now considered requirements, while others are still at the research stage. The issues, however, continue to stimulate further investigation and improvement.

3.5. Research Applications:

The few research applications of mobile-agents quoted by different authors in various domains are stated below:

- NASA’s news dated 29th April 2004 States “Playing the role of astronauts, researchers carry ‘smart’ laptop computers that talk with a prototype robot during the tests. The laptops are loaded with ‘mobile-agent’ software that scientists say will improve communications between human planetary explorers, robots and mission support on earth. Scientists said the robot traveled a total distance of 1,162 meters (3,835 feet) on Tuesday, April 27, a new distance record for the Boudreaux robot, and a rare event for mobile autonomous robots working in such extreme environments. Researchers also reported that the robot ran for almost six and three-quarters hours, a new endurance record for the rover”.

- One of the first European agents R&D projects was ‘Archon’ (Architecture for Cooperating Heterogeneous Online Systems), launched in 1989 to develop multi-agent control applications. Deployed in Iberdrola, Spain, Archon manages power distribution in part of Spain’s electricity distribution network.

- The various applications as reported by Carles Sierra et al in their paper “Agent Research and Development in Europe”, published in IEEE - Internet Computing sept-oct 2000, are: Electronic commerce (CASBA and AIMedia), Learning
assistant agents (Comris), Geographical information systems (Agent), and Manufacturing management (Mascada and Terpsichore).

- In the article “The Vision of Autonomic Computing” by Jeffrey O. Kephart and David M. Chess of IBM Thomas J. Watson Research Center, published by the IEEE Computer Society, January 2003, it is stated that autonomic elements will have a complex life cycle, continually carrying on multiple threads of activity, and continually sensing and responding to the environment in which they are situated. Autonomy, proactivity and goal-directed interactions with their environment are distinguishing characteristics of software agents. Viewing autonomic elements as agents and autonomic systems as multi agent system makes it clear that agent-oriented architectural concepts will be critically important.

- To support the constant access of ubiquitous applications, Marcela Rodriguez et al, in their article “Autonomous Agents as conference Aids in Ubiquitous Collaborative Environment”, have proposed the use of autonomous agents to which a user delegates responsibilities while being disconnected from the network.

- The paper “Ubiquitous Application Development using a Mobile-Agent based System”, by Kazutaka Matsuzaki et al, University of Tokyo, Japan, introduces a workflow-awareness model based on agent pairing which makes it possible to tune up a performance of the application without disorganizing the application logic.

- Rebecca Montanari et al, in their paper “A Context centric access control middleware for Mobile environment”, in Mobile-Agents for Telecommunication Applications, 5th International Workshop, MATA 2003, Vol. 2881 of Lecture Notes in Computer Science, proposes the use of a mobile-agent technology, where, mobile-agents are deployed in a wired network and works as proxies in using services for mobile user who requests to use the services.
• Multi-Agent System based Autonomic computing environment was proposed by Jun Hu et al, in the third International Conference on Machine Learning and Cybernetic, Shanghai, August 2004 IEEE.

• The Everything is Alive (EiA) project at the University of Arkansas developing an agent system for pervasive computing- extending Internet and Web-based agents to communicate with everyday things that, in turn, communicate with every other. In this world, toys play together, pets converse with their owners, vehicle talk to road signs, refrigerators know when item inside expire.

• Data mining algorithms and their products have been designed with two wrong assumptions. The first is that data are static and available at single location in all aspects. The second problem is, they assume that execution environments with abundant resources exist. Kok-Leong Ong et al, in their paper (most recent compared to our work) "Agents and Stream Data-Mining: A New Perspective", IEEE Intelligent systems-2005, propose the use of agents for data mining.

• The article "Agents and Markets" by Amy Greenwald et al, in the guest editors introduction column of IEEE Intelligent Systems-2003, states that coupling of autonomous agents and market places will fundamentally change how many goods and services are traded and also establishing trading relationship more timely fashion.

• Designing agents that can bid in online simultaneous auctions is a complex task. Amy Greenwald et al, in their paper "Autonomous Bidding Agents in the Trading Agent Competition", IEEE Internet Computing 2001, describes task-specific details and strategies of agents in a trading agent competition.

• Manufacturing is an inherently distributed process that requires effective management of physically constrained resources. Albert D. Baker et al, in their paper "Agents and the Internet: Infrastructure for Mass Customization", describes the "AARIA- Autonomous Agents at Rock Island Arsenal" project which explores how the agent-technology can combine the inherent distribution with the
internet's global communications infrastructure to make virtual manufacturing more cost-effective than existing, centrally managed operations.

- 'Nomad' is the mobile-agent system integrated with eAuctionHouse as proposed by Tuomas Sandholm et al, in their paper "Nomad: Mobile Agent System for an Internet-Based Auction House", at IEEE Internet Computing 2000. Here the mobile-agents travel to the eAuctionHouse site and participate in auctions on the user's behalf. Users can create agents using Java or can automatically generate agents from Nomad's template agent library.

- The article "Sensors + Agents + Networks = Aware Agents", by Michael N. Huhns in IEEE Internet Computing 2000, emphasis on: "Agent Must be aware of each other" and "Agents should be aware of its physical environment" to make Aware Agents in order to take the true benefits of agents in building applications.

- The Internet provides a large-scale environment for software agents. Agents are autonomous (Mobile) processes, capable of communication with other agents, interaction with the world, and adaptation to changes in their environment. N.J.E. Wijngaards et al, in their paper "Supporting Internet Scale Multiagent Systems", Data & Knowledge Engineering, 2002 Elsevier Science, propose AgentScape, a scalable agent-based distributed system.

Most of the organization have Computer network and managing these becomes cumbersome with traditional approach. The various users load their own software in individual desktop, which comes with enormous amount of memory, many times not being used properly. Many of the machines are not being utilized properly. There has to be automatic mechanism to extract resource available in network, knowledge about their usage and usage patterns, for the administrator to know what is happening and how things are being used, for planning the future. Network administrators are extremely interested in the questions like:

- How are people using computing facilities?
- What all the softwares loaded on each machine?
- How are software being used?
- Details like machine down time, usability etc....

Network administrators use traditional methods and tools to maintain the systems in network. However the effectiveness of these methods and tools are not satisfactory. Existing approaches are not capable of discovering the resource and knowledge automatically from the network transactions.

Data mining is the process of finding valuable business information in a large database. Given databases of sufficient size and quality, data-mining technology can generate new business opportunities by providing following capabilities:

- Automated prediction of trends and behaviors.
- Automated discovery of previously unknown patterns.

Web/network usage data collected in various ways. There is need to preprocess the data [1] to make it easier to mine for the knowledge. Clearly improved data quality [1] can improve the quality of any analysis on it. Usage data collection on the web/network is distributed by its very nature. If all the data were to be integrated before mining a lot of valuable information could be extracted.
A mobile-agent is a running program that can move from host to host in a network when and where it chooses [13,14,15]. Mobile-agents are one form of mobile-code. In its simplest form, the concept of mobile-code involves dynamically installing a code on a remote host. In Web applications, applets and servlets are a common form of mobile code. The mobile-code concept also appears in systems that extend the notion of remote procedure calls to transport the procedure to the server along with the call, remote evaluation. Many researchers extend the mobile-code concept to, in which an object (code and data) moves from one host to another. The mobile-agent abstraction extends this notion further by moving the code, data, and a thread from one host to another. A mobile-agent runs in one location, moves to another host, and continues at that host. An external agent usually moves mobile-code and mobile-objects; mobile-agents usually have migration autonomy. Mobile-agents offer many potential advantages over traditional approaches [14]. Compared to more traditional client server approaches, mobile agents can avoid transmitting a large amount of data across the network, which is of particular value when the network is slow or unreliable. The mobile-agent can move, with partial results, from one server to another until it has accomplished its task, then return to the originating host. In addition to speed and reliability improvements, mobile agents can also help in structuring distributed applications. When mobile-agents travel from one system to another in a network, they transfer their code, data, and execution state. Because they access systems locally, transferring multiple requests and responses across congested network link is not necessary, thus making overall performance more efficient. Consequently mobile-agents create new paradigm for data exchange and resource sharing in rapidly growing and continually changing computer networks. The mobile-code promises to increase system flexibility, scalability, and reliability. To date, however, this promise has been only partially fulfilled. Among the reasons for the technology's unmet potentials are security concerns and incomplete knowledge of the possible consequences of mobile code use [11].
4.1. Research Issues in Mobile Agents:

Several mobile agent researchers gathered to discuss the future directions of mobile-agent research. The event was the most recent in a series of Dartmouth Workshops in Transportable Agents and was held in Zurich immediately following the joint conference on Agent Systems and Applications and Mobile-Agents in September 2000 (Refer appendix-A for the details of researchers participated). The article [4] represents an amalgamation of the comments made during the discussion. The purpose of their discussion was to identify key research directions that will let the mobile-agent research have an impact, beyond the immediate mobile-agent research community, into other research areas in computer science and the commercial world. Few summaries of those discussions [4] are as:

"There is an unfortunate tendency to reinvent ideas from earlier research in distributed computing, relabel the ideas in a mobile computing or mobile agent context, and republish. We should embrace ideas from other fields and identify which ones are useful in the context of mobility, but we should avoid "reinventing" these ideas. We should also avoid overselling mobility's value. Mobility is useful in some but not all situations. Our research's goal should be to help the broader community understand when, and how much, mobility might be of use. Recent years have seen the development of many mobile agent systems based on several slightly different semantics for mobility, security, and communication. The community now needs to start distilling the best of these ideas from all of the proposed approaches and identifying the situations where those approaches best apply. We need quantitative measurements of the value of each form of mobility, communication, and so forth. We need analysis of the value of these ideas in helping structure distributed applications. Specifically, when, where, and why are different forms of mobility useful?"

Distributed system employ models in which processes are statically attached to hosts and communicate by asynchronous messages or synchronous remote procedure calls. The mobile-agent technology extends this model by including mobile processes, which autonomously migrate to new host. Widespread acceptance of mobile-agents is hampered by insufficient security mechanism. Mobile-agents present new security challenges because existing security policies are inadequate. Current policies and administration practices implicitly assume a tightly coupled relationship between where a program is stored and where it is executed. Decoupling a program's storage and execution location and allowing it to migrate during execution according to its own internal logic directly
contradicts existing access-control models. This decoupling creates two entities that require protection: remote hosts and mobile-code programs that migrate to them. There are many mechanisms like Digital signatures and trust management to protect a host. Researchers have tried these several approaches to develop security mechanisms to combat mobile-agents [4,5,6,7,16]. But the issue of authenticating the agent that has migrated to different host for participating in to the business remains unaddressed.

There are many applications of mobile-agents like Network Monitoring, Resource Discovery and Allocations in Grid Computing, Network Routing, etc. Interestingly, E-market opens up an exciting world for software agents and middle agents- a market place, where automated agents can represent humans and sometimes even outperform them. Mobile-agents are also used to provide context aware services in Ubiquitous applications. In a distributed system, failure can occur in any hardware and software components. For any system, a reliable operation is attained when all components of the system work according to specification. The mobile-agent can get lost due to host failure or many network problems. Therefore reliability is a vital issue for deploying the mobile-agent system. Extensive research [35,36,37,38,39], has occurred in the areas of survivability and fault tolerance. They addresses the fault tolerance issue in mobile-agents applications and provides various solutions for the fault tolerance but never addresses the Agent tracking from the perspective of the owner of the agent, who created and migrated it to the different context for doing the various transaction and wishes to know, at any given point of time: where is the agent? , and what is the status? Most real world agent applications need some level of Agent tracking in order to ensure error-free operations. Agent-systems can be hard to debug due to their distributed asynchronous nature. Hence special attention needs to be given to diagnostic tools for agent-based systems before they are deployed in the real world.
4.2. Research Issues in Pervasive Computing:

The advances in computing and communication technologies and software have resulted in an explosive growth in computing systems and applications that impact all aspects of our life resulting into new era of network of intelligent autonomous objects. The major challenges here can be categorized as follows:

- Networking Challenges.
- Challenges in Creating Context-Aware Information.
- Ubiquitous Device Challenges in Data Management Perspective.
- Object Categorization and Identification.
- Legal Issues.

Networking Challenges [32]:

1. Integrating the entire infrastructure that is available in order to create a new ubiquitous infrastructure
2. Millions of devices all communicating with one another will exhibit many unintended behavior. A major research priority is to develop abstractions that predict the behavior of large number of interacting devices.
3. Discovery protocols power aware computing and communication, data dissemination and aggregation, replication and reliability, high-level composable (services), security and authentication.

Challenges in Creating Context-Aware Information [33]:

Context awareness at both user level and system level are as discussed below.

Users' Perspectives:

1. Adaptiveness and Personalization

Adaptiveness and personalization will continue to be a key to context-aware systems.
2. Privacy and Security:
In creating a context-aware information, the users are always concerned about the security and privacy of the information created. In ubiquitous environment, objects are accessible to the other objects without the users’ knowledge. Therefore defining privacy policy for each object is very much relevant to the user.

3. Proactiveness:
Proactiveness means to process information on behalf of a user so that an action can be taken without requiring the users’ attention i.e. knowing what a user would want to do with the requested information, and detecting patterns in their behavior.

4. Tractability:
Tractability means that a user can see why something (proactive) happened? From the ubiquitous computing point of view, it could be argued that a human should be able to know what is happening in the background, in terms of internal state and configurations of the system.

Systems Perspectives:

1. Dynamic Connection:
This is concerned with the reliability of the connection established with the sensors and the data cache issue in case of failure of the connection.

2. Tight Inter-Relationship:
This deals with the prediction of some context parameters based on the inter-relationship between inferred context and the sensed context of the system. For example, the number of computers in the room and the energy usage of this room are closely related.

3. Learning and Reasoning:
This deals with establishing new context based on the existing interrelated context. For example, from the two given context like: “The person is running” and “High pulse rate”, it may be inferred that ‘person is jogging’.
4. **Metadata about Information:**

   This is necessary for the constant data flow between the context database and the numerous sensors.

5. **Storage and Logging of Context Information:**

   Because context-awareness means to be proactive and to detect patterns according to users' behaviors, context information and related reactions need to be stored, resulting into a number of questions related to: what, where, and how to store context information.

---

**Ubiquitous Device Challenges in Data Management Perspective:**

For devices, the emphasis has been on improving the functionality in addition to reducing size, cost and power requirements.

**Categories of Functionality [24]:**

The functionalities can be classified into the following:

1. **Support for Mobility:** The compactness of the devices combined with wireless communication means that the devices can be used in mobile situations. Thus, existing applications must be able to operate in varied and dynamic communication and computation environments, possibly moving from one network or service provider to another. Furthermore, new applications that are location-centric will also be developed.

2. **Context Awareness:** If devices become truly ubiquitous, then they will be used constantly in a wide range of continually changing situations. For the devices to be truly helpful, they must be aware of the environment as well as the tasks that the user is performing or will be performing in the near future. Context aware applications range from intelligent notification systems that inform the user (hopefully) of important events or data, to "smart spaces", that is, rooms or environments that adapt based on who is present and what they are doing.

3. **Support for Collaboration:** Another key theme of ubiquitous computing applications is the support of groups of people. This support consists of communications and conferencing as well as the storage, maintenance, delivery, and presentation of shared data. Collaborations may be performed in real-time, if all of the participants are available,
or may be done asynchronously otherwise. In addition to supporting on-going collaboration, access to and analysis of traces of last activities is also required.

**Challenges in Object Categorization and Identification:**

One of the major challenges is categorizing the objects of the ubiquitous world and assigning identity to each and every object of the ubiquitous world. Everyday objects communicate with each other and also with the Internet objects available today. People have an increasing desire for such ubiquitous access to information, anywhere, anyplace, and anytime. As ubicomp occurs everywhere, there may be a very large number of users/objects interacting with each other simultaneously in future. This necessitates the question of how to uniquely provide identity to infinitely growing objects and standard towards the uniformity of the data being exchanged by the various objects. The ultimate goal [13] of ubiquitous computing is to “make it unnecessary to carry around anything with you.” In our current day society, we must carry many items of necessity and/or efficiency including:

- Watch, alarm clock
- Wallet Money, credit cards, membership cards, driver’s License, business cards, passes, coupons, receipts...
- Cell phone, paper
- Laptop computer, PDA
- (Digital) camera, audio recorder
- Document, Planner
- (Blank) paper, pen, marker, pencil, post-it note
- Keys (for houses, cars, offices, ...)

If computing becomes ubiquitous and fully connected to its full potential, there will be no need to carry these items at any time. For this it is very important to categorize the objects as ubiquitous objects and assign each and ever-ubiquitous object a unique Id.
Legal Issues [34]:

It is guessed that various Legal issues that have been not thought so far will occur in future ubiquitous information society. Therefore it seems that many kinds of legislation will be required. Some of those are discussed below:

First of all, Personal Information Protection Act should be examined. This Act puts entities handling personal information database under various obligations in order to protect personal information. But it will be not enough in the ubiquitous information society, because in that society a person who doesn’t have personal information database can create instantly information that can specify individual by collecting information scattered everywhere on the network. Consequently, it seems that some legal correspondence is necessary. Then, Legislation of Law for surveillance camera should be considered, since video cameras will be installed everywhere in future ubiquitous information society, legal regulation will be necessary. Unauthorized computer access act should be examined. Moreover, legislation about personal authentication will be needed. In the ubiquitous information society, it assumed at that people could freely exchange information and receive services anytime and anywhere. Consequently it will be important that user is surely authenticated anytime and anywhere. For personal authentication, there are some means such as ID/password, electronic signature, but they are insufficient. Personal authentication using biometric data namely “biometrics” will be important from now on. However, biometrics will cause privacy and other legal issues. Therefore various legislations will be needed for using biometrics widely.

4.3. Research Issues in Data Mining:

Progress in bar-code technology has made it possible for retail organizations to collect and store massive amounts of sales data, referred to as the basket data. A record in such data typically consists of the transaction date and the items bought in the transaction. Successful organizations view such databases as important pieces of the marketing infrastructure. They are interested in instituting information driven marketing processes,
managed by database technology that enables marketers to develop and implement customized marketing programs and strategies. Knowledge discovery and data mining deals with the problem of extracting interesting associations, classifiers, clusters, and other patterns from data. Finding all such patterns is valuable for cross marketing and attached mailing applications. Other applications include catalog design, add-on sales, store layout, and customer segmentation based on buying patterns. The databases involved in these applications are very large. The emergence of network-based computing environments has introduced a new and important dimension to this problem, that of distributed sources of data and computing. The internet, corporate intranets, sensors networks, and even scientific computing domains support this observation. The advent of laptops, palmtops, handhelds, embedded systems, and wearable computers are also making ubiquitous access to a large quantity of distributed data, a reality. Advanced analysis of distributed data for extracting useful knowledge is the next natural step in the increasingly connected world of ubiquitous and distributed computing. Most of the popular data mining algorithms are designed to work for centralized data and they often do not pay attention to resource constraints of distributed and mobile environments. Recent research in this area has demonstrated that handling these resource constraints in an optimal fashion requires a new breed of data mining algorithms and systems that are very different from their centralized counterparts.

The process of data mining is becoming harder for large datasets because of the following reasons:

- Data stored online doubles every year
- Datasets are distributed geographically
- Datasets are immovable

In traditional approach, data mining algorithms are applied to the data at a single location. When data is collected in distributed way; this means that data must be transferred to single central place to enable conventional algorithms to be applied. This approach is costly in terms of communication, storage at central site. There are various new approaches proposed in the recent past and are discussed below.
In lightweight Distributed Data-mining approach [58], predictors are locally built on each distributed data partitions and then these predictors are sent to central site, where voting combines them. A typical clustering algorithm [60] requires bringing all the data in a centralized warehouse and involves large transmission cost. Parallelization of sequential data mining algorithms [61] still looks at the efficiency parameter of algorithm on centralized data. Work on improving the performance of distributed data mining systems by using an optimal/cost efficiency strategy has been the focus of the paper [63]. The optimization technique introduced [64] is motivated by the considerations that mining dataset either locally or by moving the entire dataset to a different server. IntelliMiner [63] is client server distributed data mining system, which focuses on scheduling tasks between distributed processors.

The various systems, which implement the Parallel Data Mining techniques, are discussed below:

PADMA (PArallel Data Mining Agents) is an agent-based architecture for data mining [54]. The PADMA system makes an effort to develop a flexible system that will exploit data mining agents of the particular application in hand. Although PADMA is not specialized for any particular kind of data mining domain, the current implementation uses agents specializing in unstructured text document classification. The Main components of PADMA are:

1. Data-mining agents
2. Facilitator for coordinating the agents
3. A Web-based user interface

Data mining agents are responsible for accessing data and extracting higher-level useful information from the data. A data-mining agent specializes in performing some activity in the domain of interest. In the current implementation, data mining agents specialize in text classification. Agents work in parallel and share their information through the facilitator. The facilitator module coordinates the agents, presents information to the user interface, and provides feedbacks to the agents from the user. The Web-based user interface interactively presents data to the user and collects user instructions.

The SPIDER (Scalable, Parallel and Interactive Data Mining and Exploration at Rensselaer) project focuses on some of the key techniques in discovery-driven data
mining [56], studying both the algorithmic and systems issues. The algorithmic aspects 
involve developing new scalable, parallel and interactive algorithms that are efficient, 
disk-based, and that can handle very large databases. The techniques studied include 
associations, sequences, classification and clustering. The systems issues focus on actual 
implementation of the algorithms on a variety of sequential and parallel hardware 
platforms, including shared-memory systems (SMPs), distributed-memory systems, 
network of workstations, and a hybrid hierarchical system consisting of a cluster of SMPs 
with a fast interconnect. Key challenges include improving locality, minimizing 
synchronization and communication, finding appropriate data layout, and improving disk I/O via pre-fetching and parallel I/O techniques.

The goal of the SPIDER data-mining project is to study the design and implementation of 
a large-scale, parallel data mining system, which can manipulate data from massive 
terabyte-sized enterprise or scientific databases, regardless of whether the data is located 
centrally or is distributed globally. This research leverages high performance parallel and 
distributed techniques in all the phases of data mining, such as initial data selection, 
cleaning and pre-processing, transformation, mining task and algorithm selection and its 
application, pattern evaluation, user interactivity, management of discovered knowledge, 
and providing tight coupling between the mining engine and database/file server.

BODHI [65] has been designed which guarantees correct local and global data model 
with low network communication load. BODHI is implemented in Java; it offers message exchange and runtime environments (agent stations) for the execution of mobile agents at each local site. JAM [66] is a Java-based multi-agent system designed to be used for meta-learning Distributed Data Mining. Different learning classifiers such as Ripper, CART, ID3, C4.5, Bayes, and WEPBLS can be executed on heterogeneous (relational) databases by any JAM agent that is either residing on one site or is being imported from other peer sites in the system. Papyrus [67] is a Java-based system addressing wide-area Distributed Data Mining over clusters of heterogeneous data sites and Meta clusters. It supports different task and predictive model strategies including C4.5.
Mobile DM agents move data, intermediate results and models between clusters to perform all computation locally reducing network load, or from local sites to a central root, which produces the final results.

Common factors to all approaches is that they aim at integrating the knowledge, which is discovered out of data at different geographically distributed sites, but it is necessary to have minimum amount of network communication, and maximum of local computation.