 UNIVERSITY OF CALGARY	Course Number: SENG 609.22	Course Name: Agent-based Software Engineering
	Session: Fall, 2004	Department: Electrical and Computer Engineering
		Document Type: Project Report

Multi – Agent Dispatch System (MADS)

Adrian Cervatiuc

Dept. of Computer & Electrical Engineering

University of Calgary

2500 University Dr. NW

Calgary, Alberta

acervati@hotmail.com

1. System Specification

A fleet company needs a 24 hours x 365 days per year breakdown/road-side assistance system for its entire fleet of vehicle.

1.1. Business case.

In order to implement such a system, the fleet company needs access to a Geographic Information System (GIS) that would give the company the ability to geo-locate the breakdown, to render it on the map, to search for specific service providers and to display routes and driving directions to service the broken vehicle.

GIS services include a set of components using data layers to produce rich GIS objects such as: complex maps, routes, distances, map reports etc. The company has two options. One is to use the external services hosted by different GIS providers, and the other one is an in – house solution where the GIS database and additional servers are deployed internally. The solution chosen for the proposed system is the first one.

In the past, companies that wanted to include location/mapping functionality in their solutions had to invest in GIS tools, license data from a variety of vendors and employ developers proficient in specialized programming languages. GIS web services vendor aggregates location and mapping data into a single, subscription-based web service taking the cost of data management off the customer’s plate. And, because this solution is based on open Internet standards, such as SOAP and XML, GIS web services put

location/mapping solutions within easy grasp of any business or independent software vendor.

There are three strong reasons for using a hosted XML web service model for delivering mapping functionality. First, they rely on a vast amount of cartographic data, points of interest such as business listings and other content such as demographic information. All of this data is temporal and needs to be updated on a regular basis to keep the system current. Getting thousands of customer organizations to do the hard work of managing data updates is not efficient; it makes more sense for the platform provider to do this on their behalf.

Second, the biggest market opportunity for location/mapping functionality is as an 'ingredient' in a broad range of applications, not as the only or main feature of an application, such as a road-side assistant system. XML web services are perfect conduits for ingredient functionality. No need for big client-side install; just efficient and standardized integration of those features (through SOAP/XML) that add value to the solution. For example, using a GPS device onboard of vehicles the company can develop a location application that can track the vehicles, display the fleet location on a map, generate reports about distances each vehicle was driven, or any other useful information.

Third, a significant part of the market opportunity for XML web services is in enabling location-based services on mobile devices or different platforms such as .NET or PalmOne. This means there is a strong requirement to be able to deliver location/mapping functionality "any time, anywhere and on any device". XML web service technology is a great fit for this.

System clients can be of two types: rich, desktop clients and thin, web based clients. Desktop clients can perform complicated operations, taking a portion of the servers work load or display/process rich maps (such as: 3-D, cadastral, etc) and map reports. Thin clients use a browser to display 2-D maps, routing information, proximities or distances. The proposed solution is the last one.

1.2 System description

The proposed Multi-Agent Dispatch System (MADS) allows 24 hours x 365 days per year breakdown/road-side assistance for entire company fleet of vehicle.

MADS allows users to define multiple sets of Points of Interests (POI) – the addresses of particular service providers where the vehicles can be repaired. Also the system will allow the user to localize the breakdown and display it on the map along with closest service providers in a certain radius.

The system will allow users to calculate and display distances and driving directions between the breakdown location and different service providers. The structure of the system is described in Figure 1.

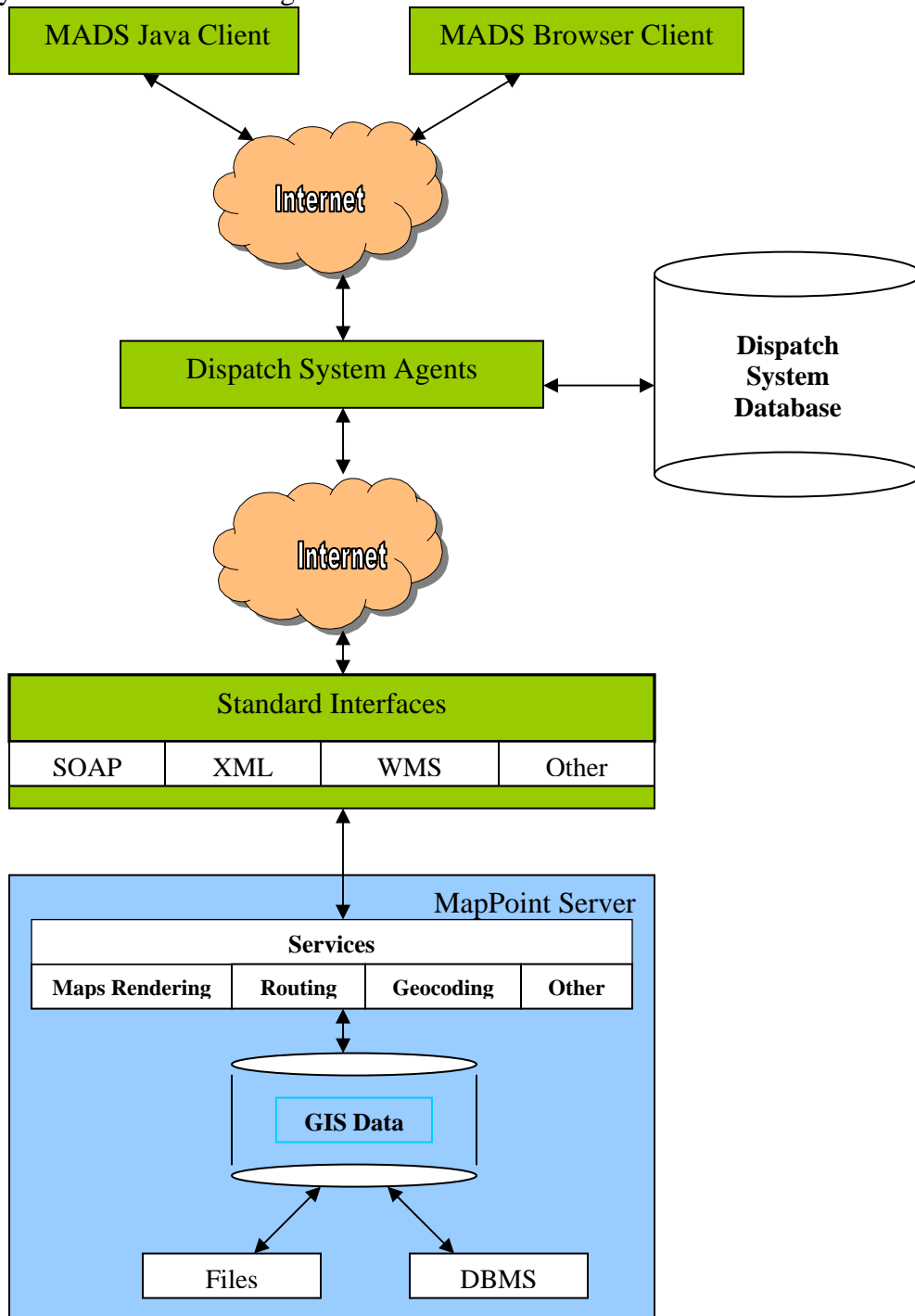


Figure 1. MADS Structure.

1.3. Assumptions

The system maintains security information for each user. It also maintains service provider information such as:

- Address and telephone
- Geocoding Information – Latitude Longitude and specific GIS information for distance calculation
- List of defects it can fix.

Each time a user adds a new service provider or modifies the address of an existing one, the system must update the GIS information associated with it through a web browser.

The solution is based on Microsoft MapPoint, a programmable location and mapping XML web service developed and hosted by Microsoft. It is registered with the Universal Discovery Description and Integration system, so that developers can find it easily, and it comes with a Web Service Description Language (WSDL) file, which means its interfaces are accessible to developers through their programming environment. It uses the Simple Open Access Protocol (SOAP) interfaces and Extensible Markup Language (or XML), which enables it to be integrated easily with other services and solutions.

MapPoint WS provides:

- Address and place finding: find street addresses anywhere in 14 Western European countries, the US and Canada, or any place name worldwide – total of 24 countries.
- Address parsing and validation.
- Map rendering: display detailed and easy-to-read maps in any application.
- Proximity searches: find particular points of interest in a specified range from a centre point.
- Driving directions.
- Tracking: track and visualize the location of moving assets, such as vehicles, on a map.
- Business intelligence: visualize specific enterprise data or standard demographic data on a map.

1.4. Requirements.

Using the MADS, the client dispatcher must first locate the vehicle. The system should provide a variety of search criteria. Examples are:

Search Criteria	Definition
Major Road Junction	Search for a major road junction
Road Name	Search by the name of the road where the breakdown has occurred. This is important if the breakdown occurs in a non

	urban area.
Motor way Junction	Search by the motorway junction nearest to the breakdown
Postal Code	Search by the postcode where the breakdown has occurred
Town Name	Search by the name of the town nearest to the breakdown. Important to locate non urban breakdowns.
County	Search for a county
OSNGR	Search by the Ordinance Survey National Grid Reference of the breakdown location
Latitude / Longitude	Search by the latitude / longitude of the location

The mapping software should also provide map manipulation functionality such as:

- Zooming in on detailed maps
- Territory planner (to allow breakdown areas to be defined)
- Multi-symbol pushpins (to allow service provider locations to be marked)
- Address and postal code validation.

Once the call is positioned on the map, MADS will allow the user to view the client breakdown areas and the client locations nearest to the breakdown. The system will ensure that users can search for a specific set of POI associated with repair providers operating in the breakdown area in accordance with the type of call that needs to be allocated. Distance and routing information need to be provided for fast response.

The decision to allocate a call to a particular service provider will be dependent upon the circumstances of the breakdown and a number of other factors. Examples are:

- Location of vehicle
- Problem reported
- Age of vehicle
- Remaining warranty on vehicle
- Whether vehicle is loaded
- Whether hazardous or perishable goods are being carried

2. System Design

Multi-Agent Dispatch System is architected as a layer of agents between the client user interface – a web browser and MapPoint existing web services.

The methodology used for the analysis and design for the Multi-Agent Dispatch System is GAIA. This methodology increasingly generates detailed models during the analysis and design phases for an agent-based system. These models are used in the design of individual agents and the communication between them. This methodology is not based on specific agent architecture; the aim of the analysis phase is to understand the internal structure of the system and the design phase focuses on the transformation of the analysis models into lower level of abstractions.

2.1 GAIA analysis.

In the analysis phases the following models are defined:

- Roles Model – a description of each role in terms of responsibilities, permissions, interaction protocols, and activities.
- Interaction model – a description of each protocol in terms of data exchanged and partners involved.

2.1.1 Role Model.

The analysis led to the following roles in the MADS: Geocode, RenderMap, Route, and PersonalAssistant.

Role: Geocode
Description : It wraps the MapPoint geocoding Web Services and it queries MapPoint for geocodes.
Protocol and Activities: <u>QueryMapPointGeocode</u> , RequestGeocode, RespondGeocode
Permissions: reads MapPoint Web Services
Responsibilities: Liveness: $GEOCODE = (GeocodeLocation)^\omega$ $GEOCODELOCATION = RequestGeocode. \underline{QueryMapPointGeocode}.$ RespondGeocode Safety: <ul style="list-style-type: none"> • a successful connection with MapPoint Web Services is established.

Role: RenderMap
Description : It wraps the MapPoint map rendering Web Services and it queries MapPoint for maps.
Protocol and Activities: <u>QueryMapPoint</u> , RequestMap, RespondMap
Permissions: reads MapPoint Web Services
Responsibilities: Liveness: $RENDERMAP = (RenderMap)^{\omega}$ $RENDERMAP = RequestMap.\underline{QueryMapPoint}.\text{RespondMap}$ Safety: <ul style="list-style-type: none"> • a successful connection with MapPoint Web Services is established.

Role: Route
Description : It wraps the MapPoint routing Web Services and it queries MapPoint for routes.
Protocol and Activities: <u>QueryMapPoint</u> , RequestRoute, RespondRoute
Permissions: reads MapPoint Web Services
Responsibilities: Liveness: $ROUTE = (FindRoute)^{\omega}$ $FINDROUTE = RequestRoute.\underline{QueryMapPoint}.\text{RespondRoute}$ Safety: <ul style="list-style-type: none"> • a successful connection with MapPoint Web Services is established.

Role: PersonalAssistant
Description : It acts on behalf of a system user. It gets the geocode information, maps and routes, and it calculates distances between the breakdown and available service providers.
Protocol and Activities: <u>LoginUser</u> , <u>CalculateDistances</u> , <u>UserGeocodeRequest</u> , <u>UserMapRequest</u> ,

<u>UserRouteRequest</u> , RequestGeocode, RespondGeocode, RequestMap, RespondMap, RequestRoute, RespondRoute
Permissions: reads userCredentials // reads the userID and password from system database
Responsibilities: Liveness: PERSONALASSITANT = <u>LoginUser</u> . ((<u>CalculateDistances</u>) ^o (ServeGeocode) ^o (ServeMap) ^o (ServeRoute) ^o) SERVEGEOCODE = <u>UserGeocodeRequest</u> .RequestGeocode.RespondGeocode SERVEMAP = <u>UserMapRequest</u> .RequestMap.RespondMap SERVERROUTE = <u>UserRouteRequest</u> .RequestRoute.RespondRoute Safety: userCredentials = true.

2.1.2 Interaction Model.

Each request protocol is described by the following attributes:

- Purpose/parameters – a brief description of the nature of the interaction and parameters used.
- Initiator – the role(s) responsible for starting the interaction.
- Receiver – the role(s) with which the initiator interacts.
- Responding protocol – the protocol invoked by the receiver in response to the initiator request.

Protocol	RequestGeocode	RequestMap	RequestRoute
Purpose/ parameters	Request to geocode a location. The response includes the Latitude and Longitude of the location	Request a map containing a list of location. The response include the URL to the map that displays the locations	Request a route between two locations. The response includes the driving direction between the locations
Initiator(s)	PersonalAssistant	PersonalAssistant	PersonalAssistant
Receiver(s)	Geocode	RenderMap	Route
Responding Protocol	RespondGeocode	RespondMap	RespondRoute

2.2. GAIA design

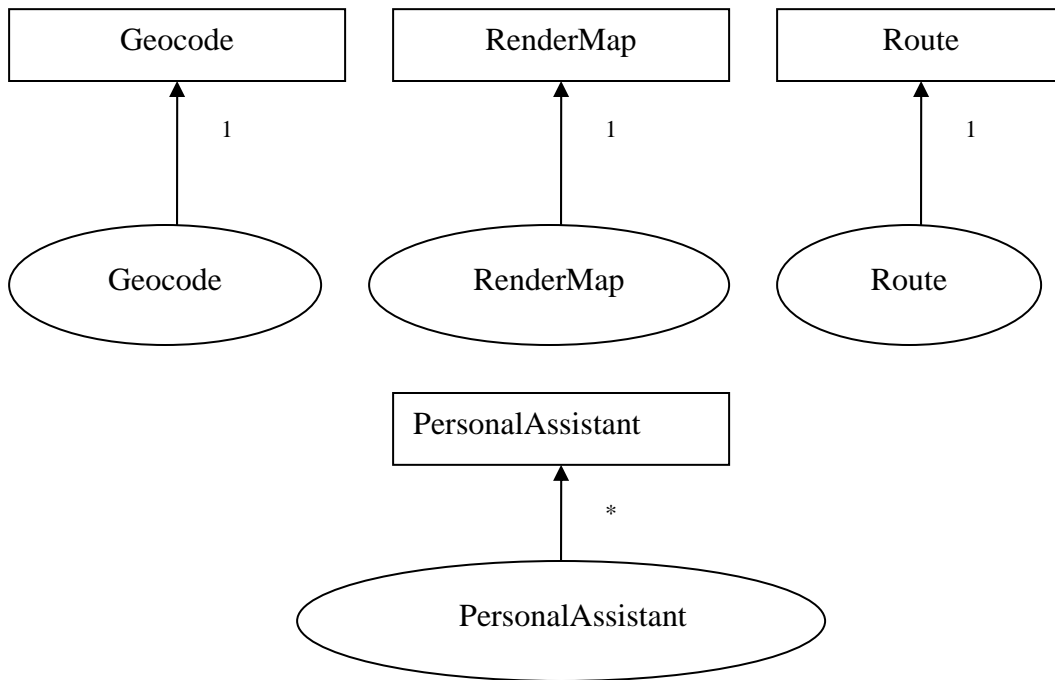
In the GAIA design phase, the following models are defined:

- Agent Model – the complex agent types and instances that compose the system; each agent is seen as a sum of agent roles.
- Service Model – services that are provided by each agent in order to fulfill its role.

- Acquaintance Model – a description of communication between different agents materialized in communication protocols.

2.2.1 Agent Model

Each emerging agent type can be represented as a role that combines all the aggregated roles attributes (activities, protocols, responsibilities and permissions). The agents model for MADS has four agent types: the geocode agent type, who fulfills the Geocode role, the render map agent type, who fulfills the RenderMap role, the route type, who fulfills the Route role, and the personal assistant type who fulfills the PersonalAssistant role. The system has only one of each geocode, render map and route agent and as many personal assistants as the users of the system. The Agent model is described in Figure 2.



Legend

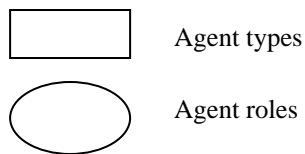


Figure 2. GAIA Model

2.2.2 Service Model

The service model lists services that agent types can provide. They are derived from the activities, protocols, responsibilities and the liveness properties of the roles. For each service, four attributes must be specified: the inputs, outputs, pre-conditions and post-conditions. The service model is described in the next table.

Service	Inputs	Outputs	Pre-conditions	Post-conditions
Geocode	Address	Latitude and Longitude of the Address	A personalized assistant agent is created and associated with the user.	User enters an address.
RenderMap	List of geocodes	URL to the map	A personalized assistant agent is created and associated with the user. The system does the proximity calculation	
Route	Breakdown and service provider location	Driving direction	A personalized assistant agent is created and associated with the user.	User selects the service provider

2.2.3 Acquaintance Model

The acquaintance model is a directed graph between agent types that allows the designer to visualize the degree of coupling between agents. An arc $A \rightarrow B$ shows the existence of communication link allowing A to send messages to B, but not necessary that B will send messages to A. MADS acquaintance model is described in Figure 3.

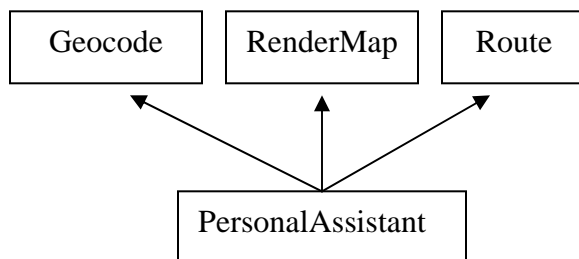


Figure 3. Acquaintance Model

2.3. Data Specification

MADS database relationship is described in Figure 4.

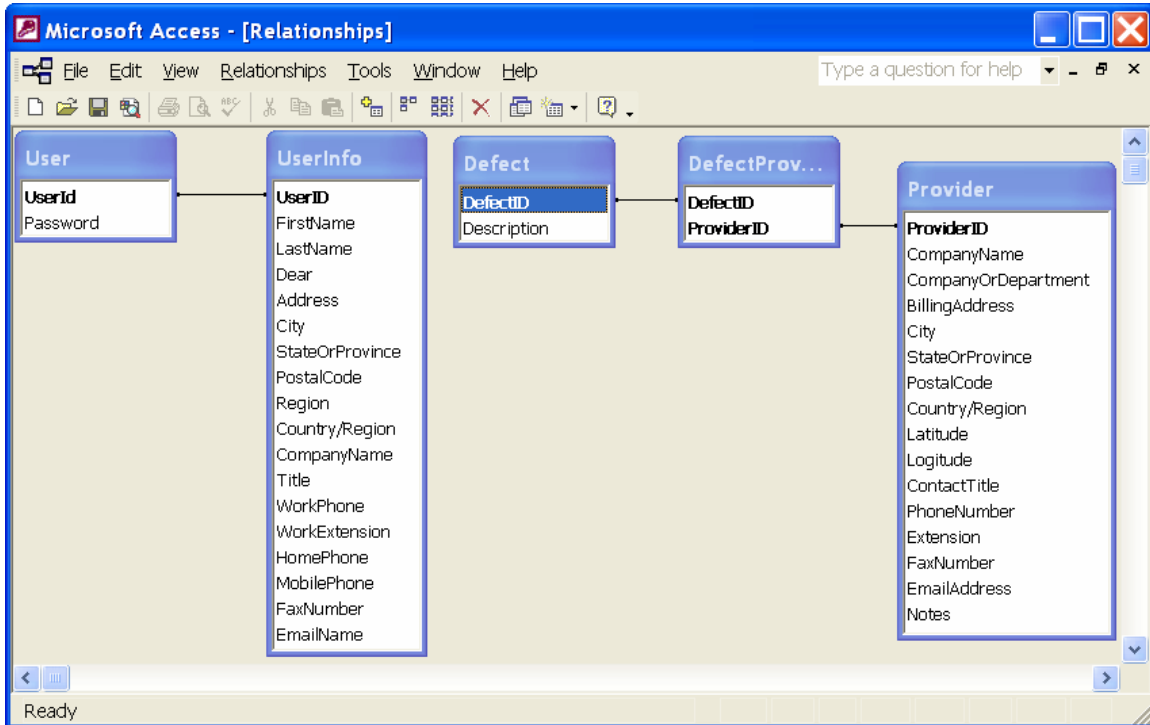


Figure 4. MADS E-R Diagram

2.3.1. Data definition

User table is used to authenticate the users in the system. UserInfo contains information related to the user. Defect table has all the defects codes and Provider table contains all the service providers in the system. DefectProvider table associates a defect with the service providers that can fix that defect. The next diagrams describe each table in the system.

Microsoft Access - [User : Table]

File Edit View Insert Tools Window Help

Field Name	Data Type	Description
UserId	AutoNumber	User Unique ID
Password	Text	User Password

Design view. F6 = Switch pan

Figure 5. User Table description

Microsoft Access

File Edit View Insert Tools Window Help Type a question for help

UserInfo : Table

Field Name	Data Type	Description
UserID	AutoNumber	User ID
FirstName	Text	First Name
LastName	Text	Last Name
Address	Text	Address
City	Text	City
StateOrProvince	Text	Province or State
PostalCode	Text	Postal Code
Region	Text	Region
Country/Region	Text	Country
CompanyName	Text	Company Name
Title	Text	Title
WorkPhone	Text	Work Phone
WorkExtension	Text	Work Extension
HomePhone	Text	Home Phone
MobilePhone	Text	Mobile Phone
FaxNumber	Text	Fax Number
EmailName	Text	Email

Design view. F6 = Switch panes. F1 = Help.

Figure 6. UserInfo Table description

Microsoft Access - [Defect : Table]

File Edit View Insert Tools Window Help

Field Name	Data Type	Description
DefectID	AutoNumber	Defect ID
Description	Text	Defect Description

Design view. F6 = Switch pane

Figure 7. Defect Table description

Microsoft Access

File Edit View Insert Tools Window Help Type a question for help

Provider : Table

Field Name	Data Type	Description
ProviderID	AutoNumber	Provider ID
CompanyName	Text	Company Name
CompanyOrDepartment	Text	Company/Department
BillingAddress	Text	Address
City	Text	City
StateOrProvince	Text	State/Province
PostalCode	Text	Postal Code
Country/Region	Text	Country/Region
Latitude	Number	Latitude coordinate
Longitude	Number	Longitude coordinate
ContactTitle	Text	Contact title
PhoneNumber	Text	Phone Number
Extension	Text	Extension
FaxNumber	Text	Fax Number
EmailAddress	Text	Email address

Design view. F6 = Switch pane

Figure 8. Provider Table description

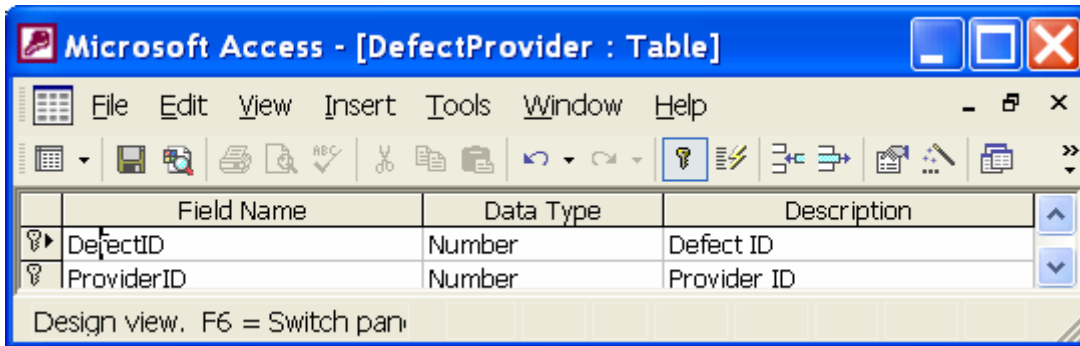


Figure 9. DefectProvider Table description

2.4 Inter-agents message format

As I described in the Design section the MADS is based on Simple Open Access Protocol (SOAP) interfaces and Extensible Markup Language (or XML).

For the Geocode service, the input parameters XML format is:

```
<address>
  <addressString>String</addressString>
  <city>String</city>
  <state/province>String</state/province>
  <postalCode>String</postalCode>
  <country>String</country>
</address>
```

For the same service the output parameters XML format is:

```
<geocode>
  <latitude>BigDecimal</latitude>
  <longitude>BigDecimal</longitude>
</geocode>
```

The RenderMap service has the following input parameters XML format:

```
<geocodes>
  <geocode>
    <latitude>BigDecimal</latitude>
    <longitude>BigDecimal</longitude>
  </geocode>
  .
  .
  .
  n
</geocodes>
```

The output parameters XML format for the RenderMap service is:

```
<mapURL>
  <url>String</url>
</ mapURL>
```

The Route service has the following input parameters XML format:

```
<route>
  <source>
    <geocode>
      <latitude>BigDecimal</latitude>
      <longitude>BigDecimal</longitude>
    </geocode>
  </source>
  <destination>
    <geocode>
      <latitude>BigDecimal</latitude>
      <longitude>BigDecimal</longitude>
    </geocode>
  </destination>
</route>
```

The output parameters format for the Route service is:

```
<route>
  < drivingDirections>
    <drivingDirection>String</ drivingDirection>
    .
    .
    .
    n
  </drivingDirections>
</route>
```