

Simulation, Planning and Operational Platform for Emergency Management Systems II (SIMPLANEMS II)



Simulation, Planning and Operational Platform for Emergency Management Systems II (SIMPLANEMS II)

Dr. Jai Asundi (asundi@cstep.in)

Dr. Dipti Deodhare (dipti@cair.drdo.in)

June 2013



Center for Study of Science, Technology and Policy (CSTEP) is a private, not-for-profit (Section 25) Research Corporation registered in 2005. CSTEP's vision is to enrich the nation with science and technology-enabled policy options for equitable growth. CSTEP's studies do not necessarily reflect the opinions of its sponsors.

Design and editing by: CSTEP

(Images courtesy)

Disclaimer

While every effort has been made for the correctness of data/information used in this report, neither the authors nor CSTEP accept any legal liability for the accuracy or inferences for the material contained in this report and for any consequences arising from the use of this material.

© 2013 Center for Study of Science, Technology and Policy

No part of this report may be disseminated or reproduced in any form (electronic or mechanical) without permission from CSTEP.

June 2013

CSTEP/0/0, 2013

Center for Study of Science, Technology and Policy
Dr. Raja Ramanna Complex, Raj Bhavan Circle,
High Grounds, Bangalore – 560 001
Tel.: +91 (80) 4249-0000
Fax: +91 (80) 2237-2619

Email: admin@cstep.in
Website: www.cstep.in

Acknowledgements

We would like to sincerely thank Prof. V.S. Arunachalam (Chairman, CSTEP) and Dr Anshu Bharadwaj (Executive Director, CSTEP) for their continuous encouragement throughout the course of this study. We are grateful to Mr. V.S. Mahalingam (Director, CAIR) and Mr. S. Burman (Director, CAIR) for support and suggestions during the course of the project.

We express our gratitude to Col. S. Sobti, Mr. R. Sri Kumar, IPS (Retd. DG and IGP), Dr. Aruna Ramesh(Head, Emergency Medicine, M. S. Ramaiah Medical Hospital), Mr. N.U. Erappa, (Chief Fire Officer, West zone) for useful inputs, feedback and contributions during this study.

We thank members of CSTEP, Dr. R. Krishnan, Dr. Annapoorna Ravichander, Bhargavi Kerur for assistance in preparing this report and Mr. Raghvendra Rau (Controller, CSTEP) , Vivek Rao, Ramanaidu M., P. Ranganathan (IT group, CSTEP) and Dr. K.C. Bellarmine (CFO, CSTEP) for their administrative assistance throughout the course of this study.

Contents

Executive Summary vii

1. Introduction..... 1

2. Artefacts and Outputs 5

3. Vehicle Dispatch and Routing System..... 12

4. Coordination Protocol Exercise 22

5. First Responder Game 36

6. 2-D to 3-D Conversion and Comparison of Game Engines for Simulation..... 43

7. Triage Exercise 54

8. Multi-Agent Models..... 63

9. Carlton Towers (A Case Study on Emergency Management)..... 73

10. Plan for Future Work 79

List of Figures

Figure 1.1: Living Lab Conceptual Diagram	1
Figure 1.2: Living Lab Architecture Diagram.....	2
Figure 2.1: Analysis of the Structure of NDMA	8
Figure 2.2: Hierarchical Structure of Institutions in NDMA	9
Figure 2.3: Functions carried out by Institutions within NDMA	9
Figure 2.4: Emergency Preparedness Map.....	10
Figure 2.5: Location for Police Stations in Bangalore	10
Figure 2.6: Location of Hospitals in Bangalore	11
Figure 2.7: Location of Fire Stations in Bangalore	11
Figure 3.1: General System Architecture (VDRS)	12
Figure 3.2: Detailed System Architecture (VDRS)	17
Figure 3.3: Application Home Page.....	18
Figure 3.4: Configuration Settings for Vehicle Types.....	18
Figure 3.5: Add/Update Fire Stations.....	19
Figure 3.6: Update Assets.....	19
Figure 3.7: Modify Configuration.....	20
Figure 3.8: Shortest Paths Determined.....	21
Figure 4.1: CPE Architecture Diagram.....	23
Figure 4.2: User Use Cases.....	24
Figure 4.3: Administrator Use Cases.....	24
Figure 4.4: CPE Class Diagram	25
Figure 4.5: Compose Screen.....	26
Figure 4.6: Inbox.....	27
Figure 4.7: View Contacts	27
Figure 4.8: View Assets.....	28
Figure 4.9: Create Exercise Screen.....	29
Figure 4.10: Sample Asset List.....	30
Figure 4.11: Sample Contacts Sheet.....	30
Figure 4.12: Create Asset Type.....	31
Figure 4.13: Create Role	31
Figure 4.14: Map/Un-map Users	32
Figure 4.15: View Message Timeline.....	33
Figure 4.16: View Summary.....	34
Figure 5.1: Rendering of a Mall Structure (Image from Maya Viewport)	36
Figure 5.2: Rendering of a Mall and Surrounding Areas (Image from Maya Viewport)	37
Figure 5.3: The Layout (Image from Maya Viewport).....	38
Figure 5.4: The Mall from a Different Angle (Image from Maya Viewport)	39
Figure 5.5: Another View of the Mall and Side Streets (Image from Maya Viewport)	39
Figure 5.6: Design Imported into CryEngine3 (for Placing Vegetation, Assets, Trees etc.)	40
Figure 5.7: A Side Street in CryEngine3 (Level Design)	40
Figure 5.8: The Side Street from another Angle	41
Figure 5.9: : CryEngine3 Material Editor.....	41
Figure 5.10: Actors Introduced via CryEngine3	42
Figure 5.11: Actor Performs and Finishes Actions.....	42
Figure 6.1: Sample Floor Plan	45
Figure 6.2: Sample Building Blocks	45

Figure 6.3: Floor Plan of a Private Building in Bangalore	46
Figure 6.4: Building Block of a Private Building in Bangalore.....	46
Figure 6.5: An Imported Model in CryENGINE3.....	49
Figure 6.6: Front View of Interiors with CryENGINE3	49
Figure 6.7: Top View of Interiors in CryENGINE3.....	50
Figure 6.8: An Imported Model in UDK.....	50
Figure 6.9: Interiors in UDK.....	51
Figure 6.10: Interiors in UDK (with more detail)	51
Figure 6.11: Flow diagram for Game Design	52
Figure 7.1: Algorithm for TRIAGE	55
Figure 7.2: Work Flow Model for Administrators and Users.....	56
Figure 7.3: Use Case Diagram for the Triage Exercise tool.....	57
Figure 7.4: Triage Domain Model-ER graph	58
Figure 7.5: General System Architecture (Triage Exercise)	59
Figure 7.6: Detailed System Architecture (Triage Exercise).....	61
Figure 7.7: Class Diagram - Controllers	61
Figure 7.8: Triage Exercise in Action (User view during an exercise)	62
Figure 8.1: Conceptual Framework for an Agent-Based Model.....	64
Figure 8.2: Detailed Framework for an Agent-Based Model.....	65
Figure 8.3: Multi-agent Model Initial View.....	71
Figure 8.4: Multi-agent Model Traffic Increase	71
Figure 8.5: Multi-agent Model Emergency Traffic Re-routing	72
Figure 8.6: Multi-agent Model Emergency Vehicles Converging.....	72
Figure 9.1: Traffic Jam on Mahatma Gandhi Road in Bangalore.....	73
Figure 9.2: Carlton Towers, Bangalore	74
Figure 9.3: Smoke Coming Out of Carlton Towers in Bangalore, 2010	75
Figure 9.4: Panic during the fire at Carlton Towers in Bangalore, 2010	76
Figure 9.5: Rescue Attempts during the Fire at Carlton Towers in Bangalore, 2010	76

List of Tables

Table 2.1: Outputs as per original proposal.....	6
Table 2.2: Description of outputs	7
Table 3.1: Sample resource requirements for a fire.....	14
Table 6.1: Comparison of UDK and CryENGINE3	53

Executive Summary

When responding to internal emergencies and disasters, developing nations face many challenges. Not only is disaster management poorly understood and practised, but there are also a number of issues that affect effective response, like shortage of resources, poor urban planning, inadequate and outdated facilities and lack of training and integrated response processes among the agencies tasked with handling disasters.

CSTEP undertook a project to develop a platform that would help model the scale and impact of disasters and provide analysis, tools and exercises to handle it. The researchers examined processes that will help refine procedures for institutional arrangements, resources and asset management. The platform envisaged in the project, called the “Living Lab”, can be the central location for the Emergency Management System where all major decisions are made. The “Living Lab” platform consists of multiple, integrated sub-systems that are designed to handle different problems, yet seamlessly fit together. Some of the tools developed with this platform in mind are:

1. Vehicle Dispatch and Routing System: This system allows efficient routing of emergency vehicles to a disaster location given the constraints on type of roads, vehicles and availability of assets.
2. A process (with tools) using game-engine technology for 2-dimension (2D) to 3 dimension (3D) transformation to allow for easy visualization and situational awareness of the target structure.
3. A proof of concept exercise to train personnel in correct and effective response, called the First Responder Game which uses sophisticated graphics and game-engine technology.
4. A training and testing exercise for first responders in the START Triage procedure
5. A training tool to examine and capture coordination and control messages amongst civil security personnel during a simulated disaster called the Coordination Protocol Exercise.
6. Real world analysis requires models that are probabilistic in nature. Many times behaviours from these real-world entities may emerge that affect our understanding of a particular problem. CSTEP developed Agent-Based Models to build a tool that allows for the study of emergent behaviour
7. Finally, CSTEP also examined a disaster (Fire in Carlton Towers in 2010) in Bangalore and analysed the response of emergency management agencies. Some policy implications and procedures that may have alleviated the situation and prevented loss of life and destruction of property are indicated.

During the course of the project, many policies and technologies were examined and several proof-of-concept tools were developed. A number of articles were published in journals/conference proceedings. The goals of the project have been achieved and it will be the endeavour of CSTEP to continue the development of ideas initiated in this project.

1. Introduction

Following the Mumbai terror attacks on November 26, 2008 (infamous as 26/11), CSTEP initiated a project on emergency and disaster management. The primary goal was to develop a platform to train personnel amongst various responding agencies for collective decision-making. It was evident from the terror episode that effective coordination between multiple agencies was essential to minimise the damage caused by such incidents.

CSTEP envisioned the development of a platform that would help model the scale of disasters, and help refine procedures for institutional arrangements and asset management. The platform is a virtual environment in which one can test and evaluate different institutional arrangements, procedures, and policies. One can also examine the standardisation of operational processes and develop games for training personnel involved. Thus, the main outputs of the project are tools, models and games that will allow:

1. exploration and analysis (Identifying problems and looking for solutions)
2. simulation (Use of software tools to create certain situations), and
3. training

for effective management of natural and man-made disasters. We have termed this platform the “Living Lab” (shown below).

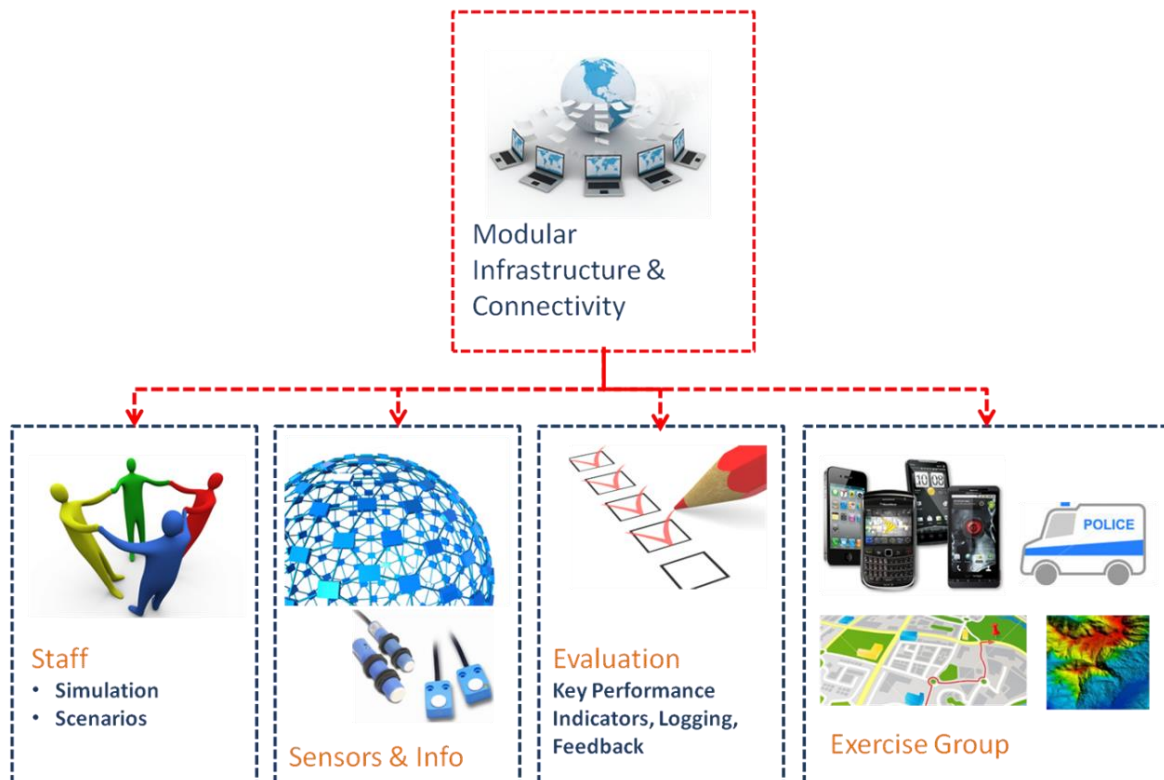


Figure 1.1: Living Lab Conceptual Diagram

This “Living Lab” is a modular platform for understanding and simulating disasters and emergency incidents. Multiple subsystems can be integrated on this platform.

In the real world, the “Living Lab” can be the central location for the Emergency Management System (EMS) where all major decisions are made. Ranking (i.e. decision-making) officials from different departments and organisations can gather to monitor and manage disastrous situations. Since it is envisioned as a web-based platform, components of the output might be remotely available to the stakeholders. This in turn will make key decisions more quick, and perhaps improve the coordinated response.

However, in order to understand the requirements of the “Living Lab” platform, it is necessary to study its functioning in conjunction with technology artefacts and plausible scenarios that may be executed within the platform. This has led to development of certain tools and exercises (in the form of games).

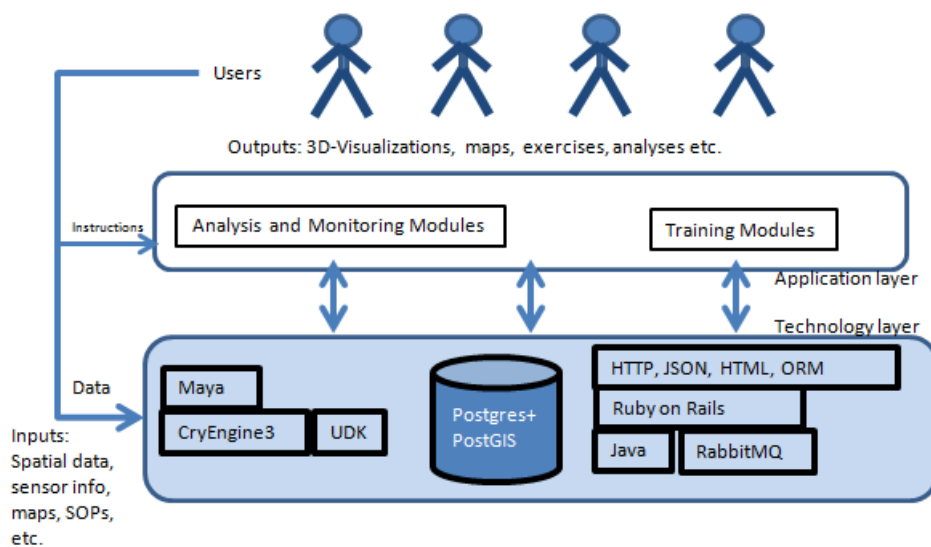
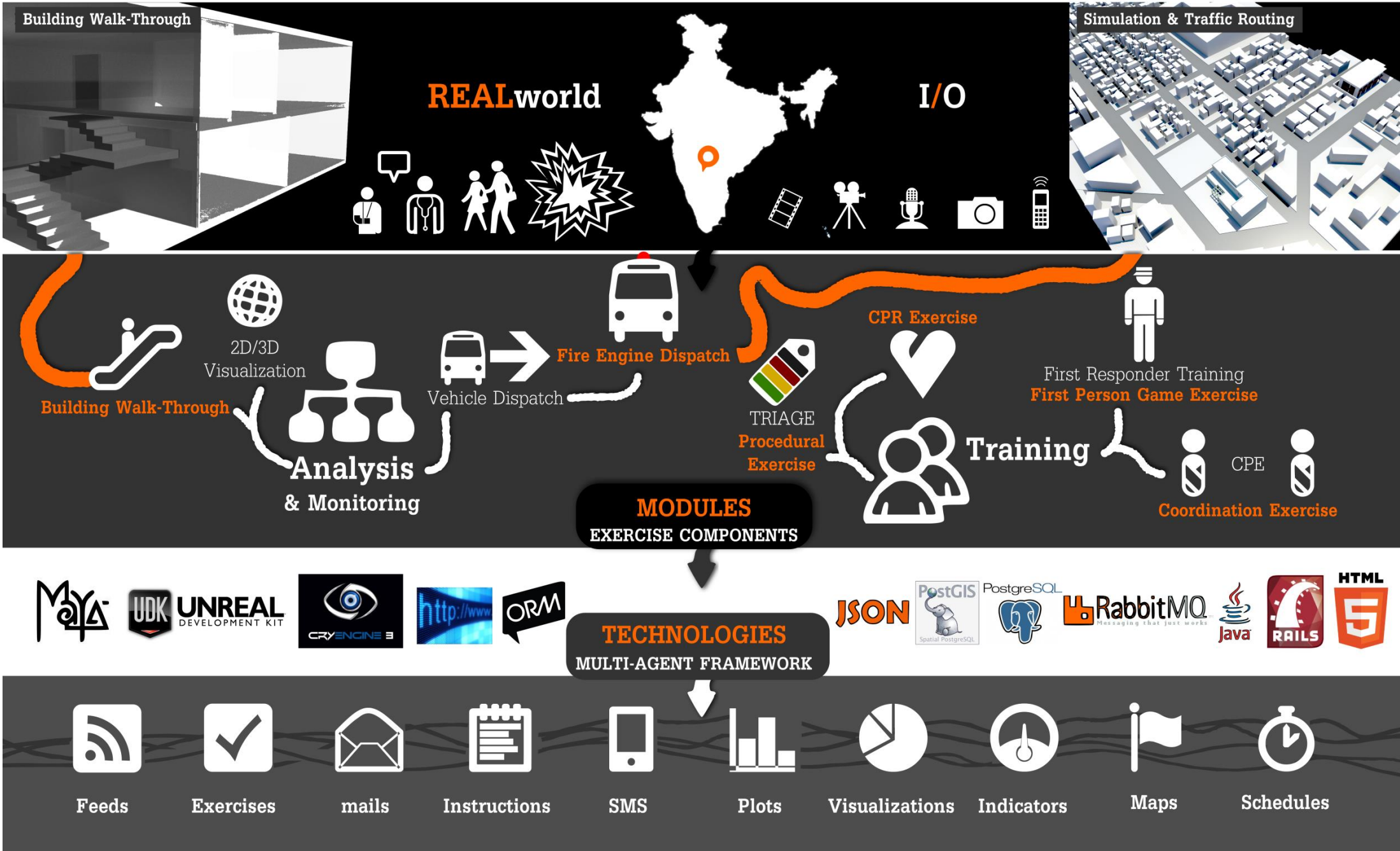


Figure 1.2: Living Lab Architecture Diagram

Expert inputs are an integral part of the domain knowledge and impetus for this research. Col. S. Sobti, Indian Army, DG & IGP R. Sri Kumar, IPS (Retd.) and currently Vigilance Commissioner with the Central Vigilance Commission, other Armed Forces and IAS personnel, and Dr. Aruna Ramesh, Head of Department, Accidents and Emergency, M. S. Ramaiah Medical College, provided vital inputs in the development of the exercises.

This report is divided into chapters to explain all the features of the SIMPLANEMS II. In Chapter 2 we briefly explain the project in detail along with a glimpse into the technology artefacts considered, used and developed. A detailed explanation of each artefact follows in the subsequent chapters. Lastly, in Chapter 9, a case study from a disaster management perspective is presented, wherein a real-life tragedy is analysed. In addition to this, some policy implications for preventing similar calamities are presented at various stages in the report.



2. Artefacts and Outputs

Introduction

The Living Lab, due to its complexity has multiple applications (or artefacts) and activities that have been developed and refined.

Artefacts

Scenario 1

Examining a fire within a building/mall: A primary requirement of first responders is access into the building and knowledge of its layout and central systems (electric wiring, water supply pipes, air conditioning ducts, fire escapes etc.). This knowledge would allow rescuers to approach the most critical sections of the structure (be it to rescue people or fight the fire) without wasting time and effort. Based on this, walk-through capabilities were developed from the 2 Dimensional (2D) building plans (given that those plans are available). This process involved the use of a gaming engine. A popular game engine (CryEngine3) is usually used to create a walk-through. The current walk-through was created in a freely available gaming engine called Unreal Development Kit (UDK) and a design software called Maya. Thus, the purpose of this exercise was twofold: (1) provide a working prototype of a needed tool, and (2) to compare the outputs of different game engines or design software and understand their relative merits.

Scenario 2

Examining dispatch of emergency response vehicles: Emergency response vehicles are usually present at multiple locations. A tool was developed that takes into account factors such as asset characteristics, road characteristics, several plausible routes etc. and suggests the optimal allocation and routes that should be taken by these emergency vehicles when responding to an incident. The initial proof-of-concept prototype is a web-based system that uses Bangalore city roads as an example.

Procedural Exercises

Computer-based procedural exercises that use a check-list based approach (similar to standard operating procedures) to train emergency responders, are developed as part of the living lab platform. The *CPR Game* and *Triage Exercise* are examples of such exercises (details follow in the subsequent chapters). The Triage Exercise was developed using the e-Adventure platform and then modified for a web-based application using the Ruby on Rails platform.

Similarly, a computer based exercise to study communication structures, hierarchy, policies and Standard Operating Procedures (SOPs) of emergency response personnel, called the Coordination Protocol Exercise [CPE], was also developed. This exercise was developed on a web-based system so that it could be administered and executed over network enabled devices.

Tables 1 and 2 list the various tasks and outputs of the project. A detailed description, including screenshots of the Living Lab follows the tables.

Table 2.1: Outputs as per original proposal

Task Number	Task	Outputs/Deliverables
1	Testing and evaluating individual models for decision making	Simulation Platform Code, Documentation, Open Source Release
2	Documents on interaction design: Interaction design for the identified scenarios and participants	Living Lab Visual Prototype
3	Scenario based integration of the living lab prototype	Living Lab Visual Prototype
4	Evaluation and testing of the system with stakeholders	Aids and Tools for the different stakeholders
5	Revised requirements for decision-making and support for the living lab	Aids and Tools for the different stakeholders
6	Additional research issues and a road map for the living lab	Living Lab Visual Prototype, Emergency & Preparedness Guide, NDMA Policy Analysis, Free Mind Maps

The outputs for the project have been categorised as follows:

Table 2.2: Description of outputs

<p>Policy Related</p>	<p>NDMA Policy Study: The National Disaster Management Authority (NDMA) is the nodal agency at the central level for disaster management in India. The agency has released a set of guidelines for disaster management in India. Free Mind maps were used to visualise and learn the organisational and operational structure of disaster management authorities in India. In addition, some points of concern in the guidelines were identified and solutions examined.</p> <p>Emergency Preparedness Guides: Based on the emergency preparedness guides created by Fire and Disaster Management Agency, Japan, after tsunami, a prototype Emergency Preparedness Guide was created for Central Bangalore. The previous maps served as iterations for this prototype.</p> <p>Asset Maps: The major hospitals, police and fire stations in Bangalore were geo-coded based on the data collection of these places.</p>
<p>Technology Oriented</p>	<p>Walkthrough of building generated from 2D building plans: A process was developed for rapid generation of a 3D walkthrough of a building based on its 2D building plans. The process involves the use of game engines as well as modelling software. The outputs from multiple processes were compared.</p> <p>Vehicle Dispatch and Routing System: This proof-of-concept tool visualises the shortest possible paths that emergency vehicles can take from source to destination subject to the constraints of vehicle limitations and road restrictions.</p> <p>Triage Exercise: This exercise is for the purpose of training medical and emergency response personnel in the TRIAGE procedure. Prototype I and II used the E-adventure game platform. This is a desktop based Java solution. Prototype III, a redesigned exercise, is a web-based system that aids the administrator to conduct this exercise from any remote location and monitor the progress of participants.</p> <p>Communication Protocol Exercise: This is a web-based system to assist in conducting an Incident Command System exercise to study the communication protocols between emergency response authorities. The earlier version of this exercise was manual.</p> <p>First Responder Game: CryEngine 3 is a gaming engine with highly realistic physics, developed by CryTek GmbH. It was used to develop a first person game to train first responders at the scene of the incident.</p> <p>Living Lab Visual Prototype: The preliminary dashboards and wire-frames have been developed for the big screens, mobiles and tablets.</p> <p>Multi-Agent Models: A new agent-based modelling platform capable of simulating large scale disasters was developed. Popular agent modelling platforms were compared and the utility of such platforms in the larger context of decision support for disaster management was analysed.</p>

Policy Study

The NDMA disaster management plan, guidelines and policies were analysed and following observations were drawn based on the analysis:

- All documents are planned with an administrative perspective, but provide little operational and engineering perspective.
- The policies direct most responsibilities down the chain of command while retaining control at the higher level. This may lead to enormous strain in the working of the institutions. A defined and easily accessible chain of command, with corresponding responsibilities, must be created.
- Many of the guidelines appear to be inspired from the existing international guidelines and no relevant references or work is shown as to how they have been adapted to the Indian context.

Free Mind Maps

A mind map is a diagrammatic structure that helps organise and display information. For example, the NDMA has an organisational structure and a set of tools that help in preparing for and responding to a disaster. A mind map of the NDMA showing different levels of authorities involved in disaster management was made using Free Mind Map software. The mapping gives the hierarchy of various disaster management authorities at National, State and District level. Also the mind map shows various tools that can/should be used by authorities as well as functions to be carried out. The following images show the mind mapping of NDMA.

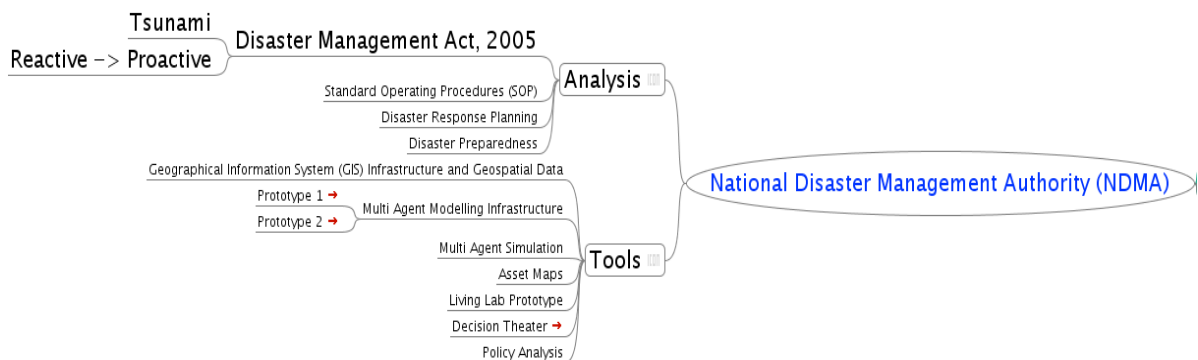


Figure 2.1: Analysis of the Structure of NDMA

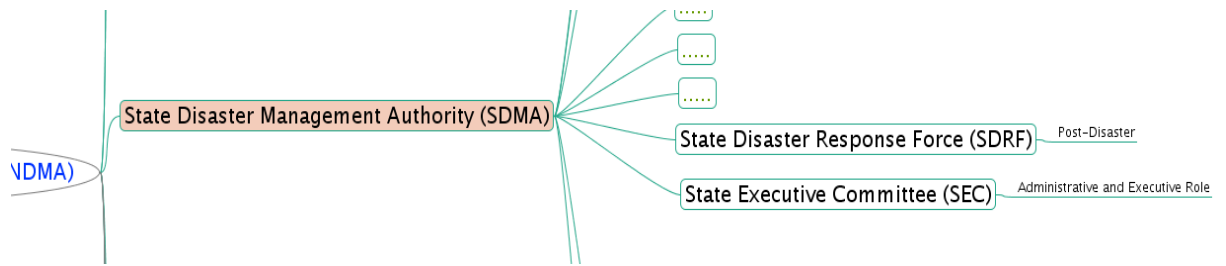


Figure 2.2: Hierarchical Structure of Institutions in NDMA

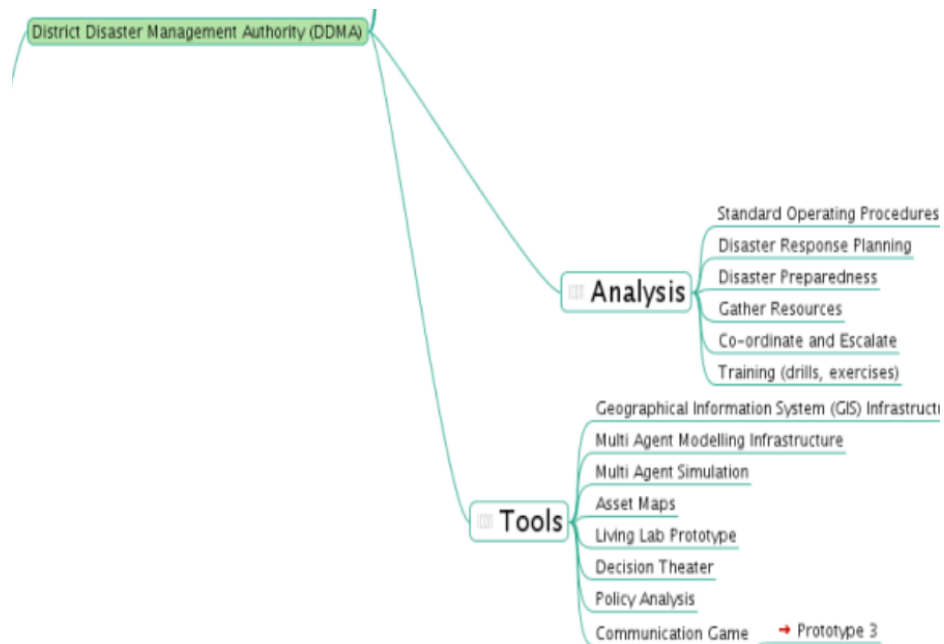


Figure 2.3: Functions carried out by Institutions within NDMA

Emergency Preparedness Guides

Based on the emergency preparedness guides that were distributed in Japan after the tsunami, a prototype Emergency Preparedness Guide was created for Central Bangalore. The previous maps served as iterations for this prototype. In Japan, these types of maps are distributed to all residents to ensure that they plan their strategy in case of an emergency. A mechanism for the production and distribution of these types of maps in India is yet to be established by any central or state agency.

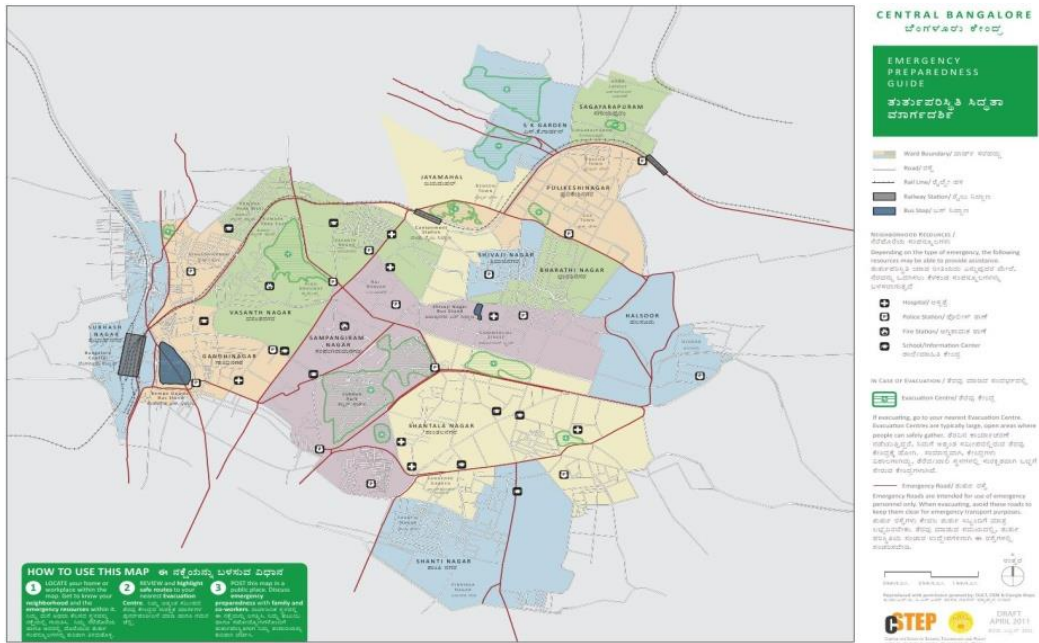


Figure 2.4: Emergency Preparedness Map

Asset Maps

We collected data on major hospitals, police stations and fire stations in Bangalore. This information was subsequently geo-coded and stored in a database. A process for the maintenance (and the updating) of this database must be created by the responsible local or state governmental body.

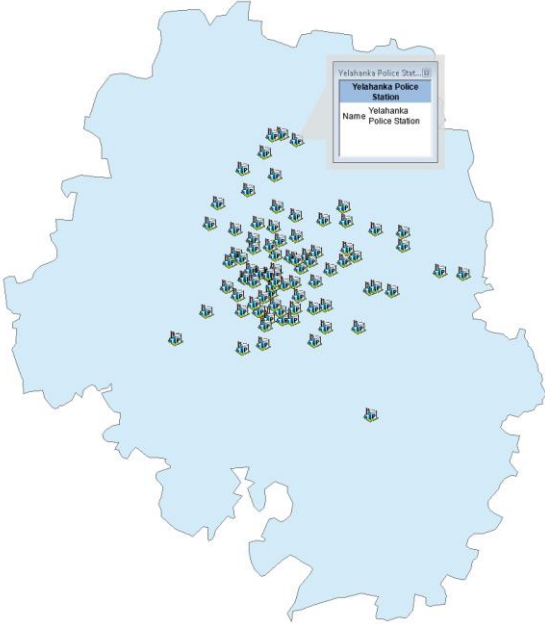


Figure 2.5: Location for Police Stations in Bangalore

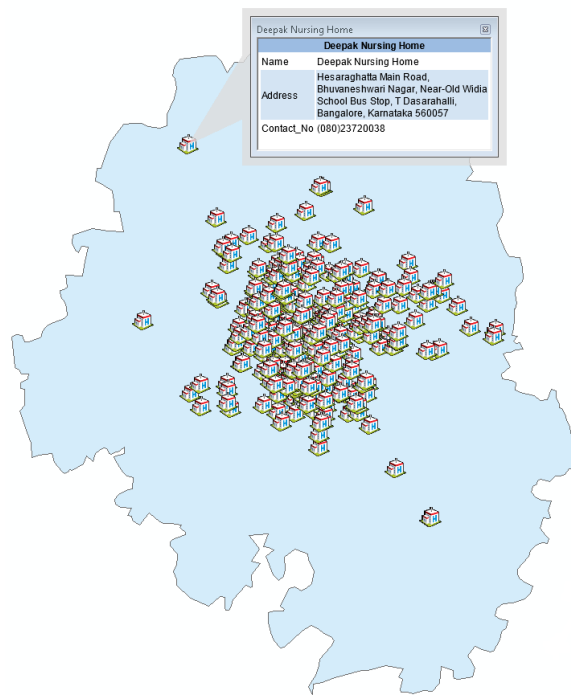


Figure 2.6: Location of Hospitals in Bangalore

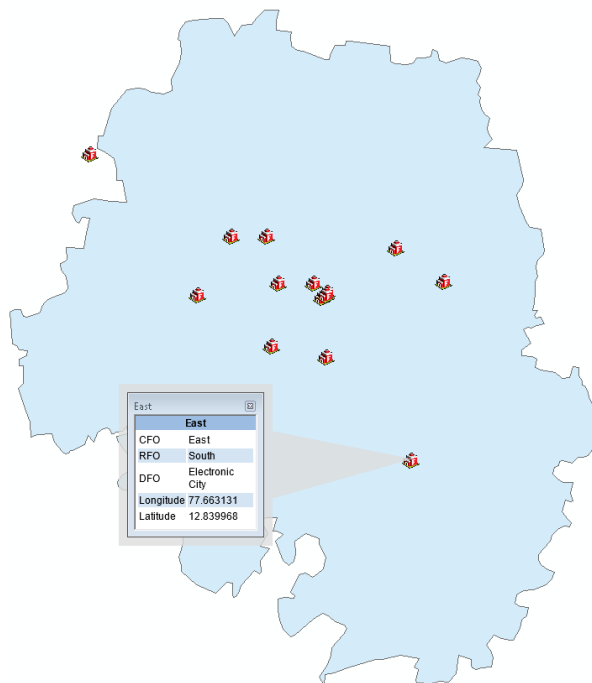


Figure 2.7: Location of Fire Stations in Bangalore

3. Vehicle Dispatch and Routing System

Introduction

Developing and underdeveloped nations have their own challenges when responding to internal emergencies. These include training for EMS personnel and macro issues like poor urban planning, inadequate and outdated facilities and lack of coordination. Responses to a disaster (like a fire or a large scale road/rail accident) are therefore slowed down by issues like narrow roads, crowd congestion and mismatched equipment (for example wide/heavy fire trucks or short ladders). Today, urban road networks lack a defined road hierarchy and associated standards. Thus, access to a site of an emergency is a problem for responding vehicles.

Apart from infrastructure and logistic problems, policies play a major role in handling these issues. In the Indian context, handling emergency situations not only requires expert and experienced manpower but also relevant data, situational reports and lists of officials of the concerned departments. Correct policies can certainly be beneficial in handling emergencies.

In this study, we have developed a visualisation tool to track the best possible paths from a resource location to an incident site. In other words, given a classification of road networks and the resource requirement, the algorithm of the tool captures the shortest path (in terms of distance) between the incident site (for example, a fire), and the resource locations (a set of fire stations) and fulfils resource requirements (types of vehicle needed based on the severity of the incident) at the site. We term it as ***Vehicle Dispatch and Routing System (VDRS)***.

System Architecture

The general Model -View -Container (MVC) architecture for the application is shown below:

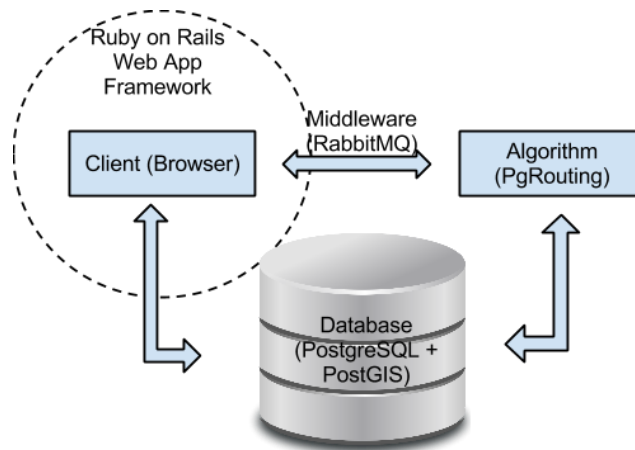


Figure 3.1: General System Architecture (VDRS)

Road Network Classification

1. Agencies such as the Federal Highway Administration (USA) and the Department for Transport (UK) provide road classifications based on certain parameters (width, load bearing capacity, direction, usage, etc.). In this application, all roads in a relevant geographical area are assigned road types based on similar parameters. The entire road network table comprises of these road types.
2. Agencies such as the Motor Vehicles Department, Govt. of Maharashtra, classify vehicle based on parameters such as weight, width, height, turn-radius etc. We define vehicle types in a similar manner. A table/map on the suitability of a vehicle type for a particular road type is then made. Based on this mapping, road networks for the movement of these vehicles are created.

Technical Requirements

The requirements are as follows:

1. A road network table with various road types (mentioned) and corresponding shape file(s) [A shape file is a geospatial data format for geographic information system software]
2. Coordinate system of these shape file(s)
3. Resource locations (for example, fire stations) and a table showing resources available at each of location

The application creates a graph from the given shape file, wherein roads are treated as edges and intersections of roads or the end points of any edges are treated as vertices. The length of an edge is proportional to the distance between the two points (or locations) that the edge connects.

Application Workflow and Components

Emergency (or Event)

A user inputs the incident location along with the requirements of assets and gets the shortest paths to reach the site from resource locations. These paths will be displayed on a map. Estimated Time of Arrival (ETA) is calculated based on the distance to be covered. Each path will be assigned a colour code based on the ETA:

- if ETA is less than 15 min, the path is shown in green colour
- if ETA is in between 15 to 30 min, the path is shown in blue colour
- if ETA is greater than 30 min, the path is shown in red colour

(Note: In the current version of the tool, we calculate the time based on a sample average vehicle speed of 5 m/s. This is an arbitrary figure and can be changed based on traffic analysis data and real speeds over specific roads.)

Algorithm

This application uses Dijkstra's shortest path algorithm. It calculates shortest distances from resource locations to the incident site based on the sum of the length of the edges along the

path from source to destination. The procedure devised here also takes the input requirements of assets into account in finding shortest paths. The algorithm is explained in detail in the section titled ‘*Working Principle of Algorithm*’.

Rabbit MQ

RabbitMQ is open source message broker software (i.e., message-oriented middleware). It is a complete and reliable enterprise messaging system based on the emerging AMQP (Advanced Message Queuing Protocol) standard. It is used to communicate between the application and the algorithm. This is based on simple Queue push-pop mechanism. The application can send the input to the algorithm and get the results from the algorithm by using RabbitMQ.

Database

The Database of the application is in PostgreSQL, an object-relational database management system. PostGIS, an open source software program, which adds support for geographic objects to the PostgreSQL is also used. The pgRouting library extends the database to provide geospatial routing functionality. Dijkstra’s shortest path function is available in the pgRouting library. The data with respect to each scenario/simulation can be saved. A detailed schematic of the architecture is provided in Appendix I.

Working Principle of the Algorithm:

Input(s) to the Application:

1. Incident Location
2. Requirements of various resource types, for example (A, 0), (B, 1), (C, 4) etc.

Table 3.1: Sample resource requirements for a fire

Resource (Vechicle) Types	Quantity required	Road Types Supported
B	1	1,2
C	2	1,2,3

(Here, the 3rd column indicates the different road types on which a particular vehicle type can move).

The pgRouting algorithm works as follows:

1. Fetches the closest vertex on the road geometry from the incident location and all the resource locations
2. Fetches the shortest path between the closest vertices considered for incident and resource locations respectively
3. Allocates the resources required by fetching them from individual resource locations, while routing on the road networks to fetch the shortest path
4. If any of the resource locations runs out of resources, it hops to the next closest resource location
5. The routing takes place on each of the road networks as defined in the configuration

Output(s) of the application:

1. Displays the shortest paths from various resource locations to incident location.

Road Network Data

1. The latest (updated) road network data were downloaded from <http://download.geofabrik.de/asia/india.html>.
2. Bangalore's road network data were extracted from the road network data collected from the link mentioned above in the .osm format.
3. An open source plugin – osm2pgrouting was used to correct the network and imported in POSTGRESQL DB.
4. The road network classification was made from the corrected network table(s) in the database.
5. This classified road network data were then converted into the *Web Mercator Coordinate System* as the Open Layers API expects the map in the same.

ETA Calculation

The following steps were used to measure the distances in metres of multiple road geometries in the application:

1. The available shape files are in the Web Mercator (SRID: 900913) Coordinate System.
2. The tables were loaded in PostgreSQL DB through the shp2pgsql-gui plug-in.
3. To measure linear distance in metres, convert the Web Mercator that are given in (Projected Coordinate System) into UTM (Universal Transverse Mercator - Projected Coordinate System).
4. Under UTM, Bangalore falls under 43^o N (N represents Northern hemisphere which can also be referred as 43N UTM zone).
5. The SRID (Spatial Reference ID) for UTM 43N zone and Web Mercator projection system are 32643 and 900913 respectively.
6. Thus, during transformation, the relevant SRID needs to be mentioned to identify the coordinate systems.
7. After the above conversion, the transformed geometry is achieved.
8. 2D length of the transformed geometry is fetched, which returns the distance of that geometry in meters.

Algorithm Summary

It works by identifying the resource locations closest to the incident based on “distances”. Once the closest resource location is known, the algorithm fetches the resources as per the requirements and identifies the shortest path from source (resource location) to the destination (incident site). If the resources are not sufficient, the algorithm chooses the next closest resource location and repeats the procedure. This continues until the requirements are satisfied. On the visualisation part, the tool displays the best possible paths from incident site to various resource locations. These paths are distinguished by colours representing various distances to enable easy visualisation. The tool also facilitates viewing the paths by selecting road networks. It can also be extended to incorporate real-time traffic analysis.

Future Enhancements

The policy implications of this study are manifold. First, it gives departments tasked with responding to emergencies and city planners an idea to optimally allocate resources by running simulations of various scenarios. Second, it could help institute a local body or state level initiative to standardise road networks (for example road widths and surfaces). Lastly, it helps in deciding tactical (on the ground) SOPs (for example, how to divert traffic away from an incident without cascading effects).

Bibliography

- Hierarchy of Roads :: <http://bit.ly/SWwRTS>
- Street Hierarchy :: <http://bit.ly/97GOeo>
- Pavement Management :: <http://bit.ly/ZhhtQy>
- Geographic Coordinate System :: <http://bit.ly/2g5Y>
- Projected Coordinate System :: <http://ibm.co/12zBqWM>
- SRID :: <http://bit.ly/10M7La4>
- EPSG :: <http://www.epsg.org/>

Appendix to Vehicle Dispatch and Routing System

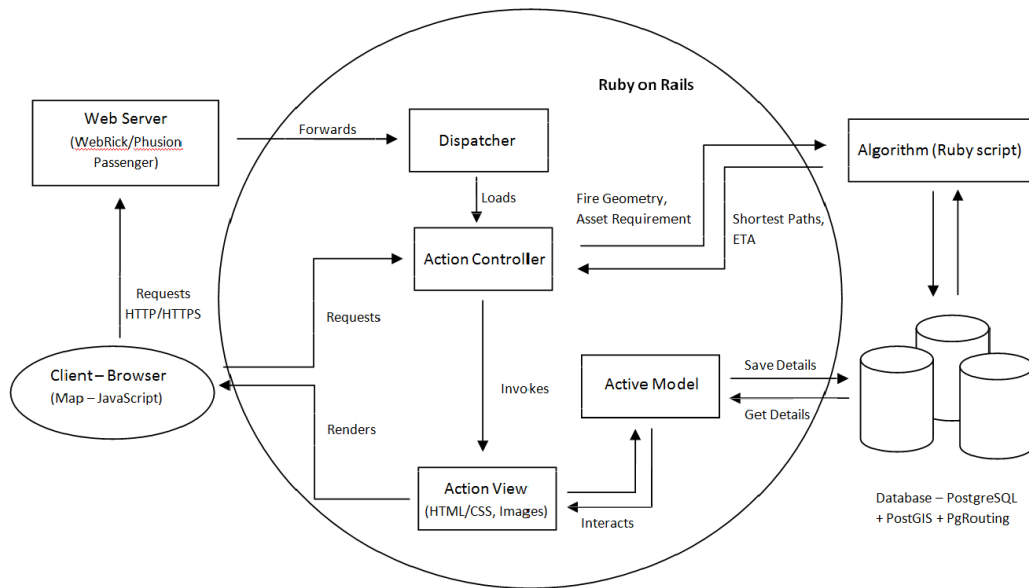


Figure 3.2: Detailed System Architecture (VDRS)

Technical Concepts Used in the Application

- UTM(Projected Coordinate System)
 - This Projected Coordinate System is a flat, 2D representation of the Earth. For more details, please refer <http://bit.ly/OgdRS>.
- Web Mercator(Projected Coordinate System)
 - Web Mercator is a Projected Coordinate System
 - Currently, there is an officially registered EPSG code 3857 whose projection is identical to EPSG:900913. So, if you need to combine overlay layers that are using either an alias or the official EPSG code with an OpenLayers Spherical Mercator layer, you have to make sure that OpenLayers requests EPSG:3857 or other alias instead of EPSG:900913.
 - For more details, please refer <http://bit.ly/edNnNb>

Special POSTGIS functions used

- ST_Distance :: <http://bit.ly/Y7xR6S>
- ST_GeomFromText :: <http://bit.ly/10Mdblg>
- ST_AsText :: <http://bit.ly/16Hp0i0>
- ST_Union :: <http://bit.ly/ZcdoOw>
- ST_MakeLine :: <http://bit.ly/Zcdqpb>
- ST_Transform :: <http://bit.ly/YFFvJE>
- ST_Length :: <http://bit.ly/144D47s>

Application screen shots and details on functional modules

1. Home Page

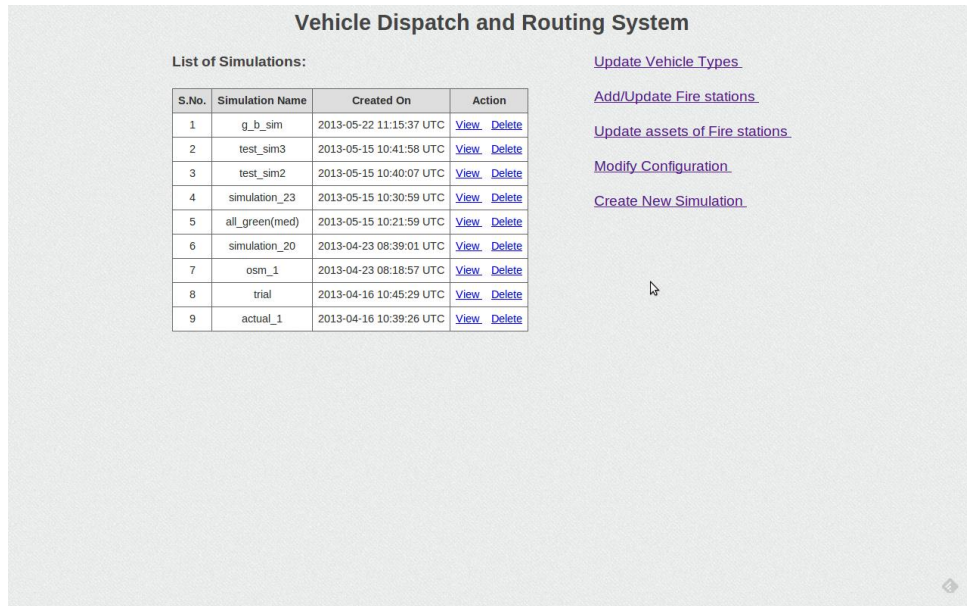


Figure 3.3: Application Home Page

2. Add, Update or Delete Vehicle Types

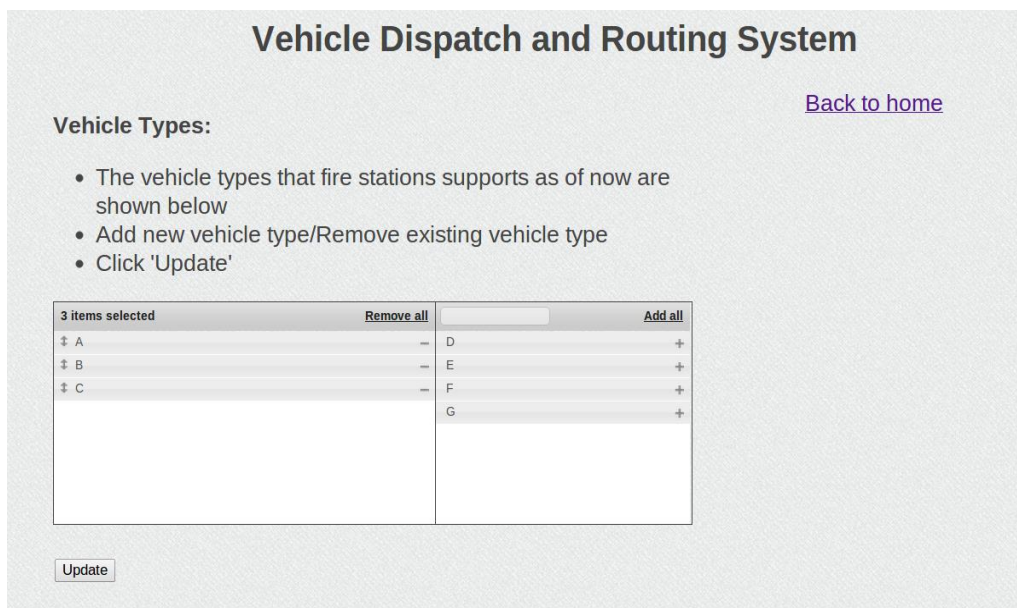


Figure 3.4: Configuration Settings for Vehicle Types

3. Add/Update Fire stations

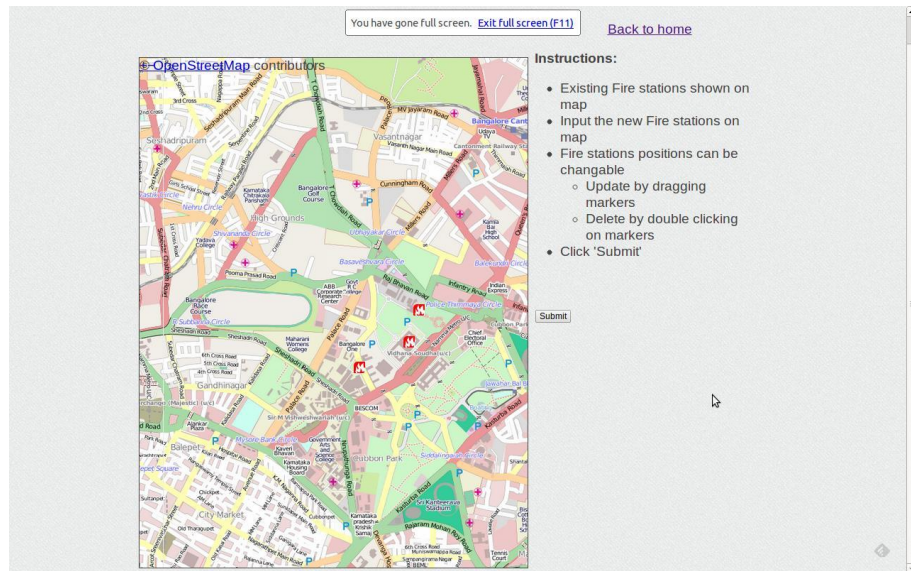


Figure 3.5: Add/Update Fire Stations

User can add new fire stations by clicking on the map. User will be able to update existing fire station locations.

4. Update assets of Fire stations

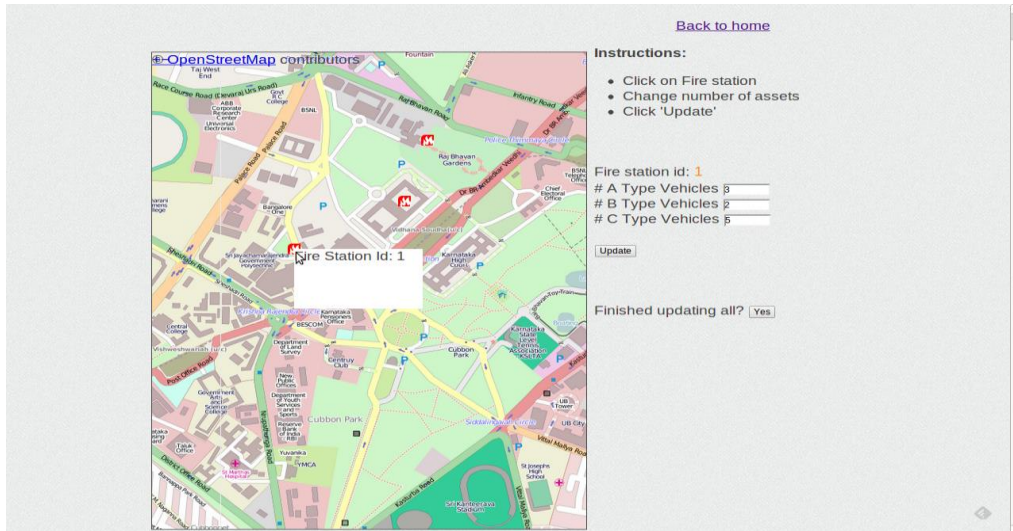


Figure 3.6: Update Assets

User can update the number of vehicles of each fire station. User will be shown current assets of fire stations and will be able to update it.

5. Modify Configuration

User can modify configuration between vehicle type to road network. Current configuration will be shown in the screen and user will be able to update it. The vehicle types for which configuration is not yet defined will be highlighted.

Current Configuration:

Vehicle Type	Can move on Road Types		
	1	2	3
A	Yes	No	No
B	No	Yes	No
C	No	No	Yes

New Configuration:

Vehicle Type	Can move on Road Types		
	1	2	3
A	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
B	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
C	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
D	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 3.7: Modify Configuration

6. Create Simulation

The user can create a new simulation by providing the number of assets required and the location of fire. He/she will be shown the shortest paths from fire stations. The paths will be displayed with different colours based on ETA. The simulation data can be saved. A simulation with the following details is shown in Figure 3.8.

Fire:

- Latitude: 1456746.4009613
- Longitude: 8638057.5623184
- Area: British Library, Bangalore

1. Fire station 3:

- Area: Vidhan Soudha
- Longitude: 77.59095300000
- Latitude: 12.97999000000
- Resources Available: A,2,B,6,C,0
- Fire vehicle of type B follows: Kasturba Road
- Kasturba Road classification: Road type 2

2. Fire station 2:

- Area: Raj Bhavan
- Longitude: 77.59165400000
- Latitude: 12.98214000000
- Resources Available: A,1,B,1,C,5
- Fire vehicle of type C follows: Vittal Mallya Road
- Vittal Mallya Road classification: Road type 3

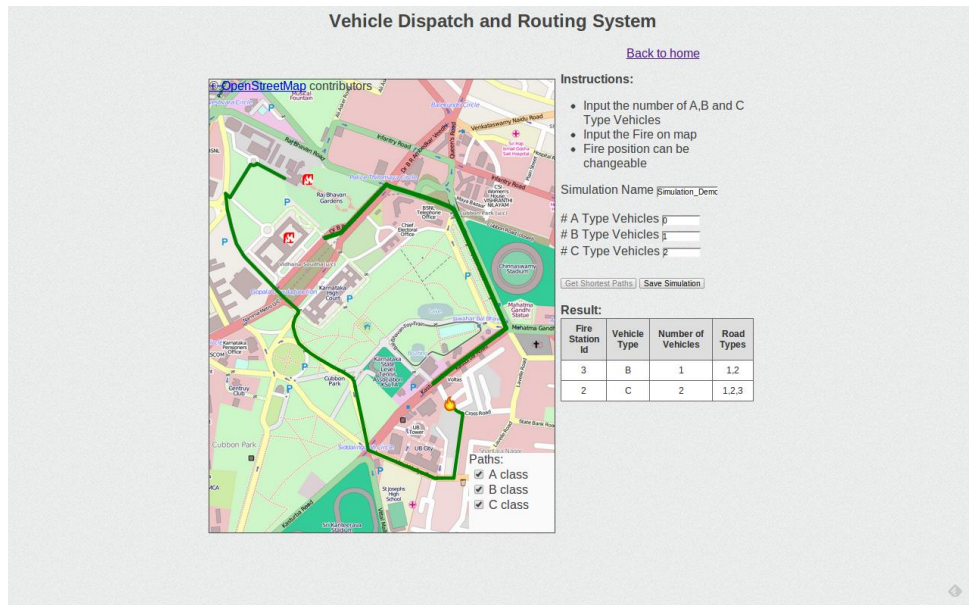


Figure 3.8: Shortest Paths Determined

4. Coordination Protocol Exercise

Introduction

Communication and coordination are vital parts of an emergency response system. CSTEP has developed a Communication Protocol Exercise (CPE) framework, where expected participants are assigned roles (and provided with assets) during an emergency. A situation is created and communicated to participants. These participants coordinate with others via message-transfer and respond to the situation collectively. The CPE tool is web-based and will also be available on mobile devices.

Background

CSTEP conducted a manual exercise that simulated an emergency situation in a city/town at the Administrative Training Institute in Mysore in a course on Disaster Management. The participants consisted of Army personnel and IAS and IPS officers. The exercise brought to light many procedural as well as structural issues.

Around 25 members participated in this workshop. To begin with, participants were assigned roles such as Commanding Officers (CO) from Police Fire, Medical, and as District Commissioner. The players were introduced to concepts, ground rules and important mechanisms of the game. Written materials and forms for communication were distributed to them and a game clock was started and displayed on the screen.

The facilitators fed a number of public information messages to certain players (by role) so that they were made aware of the situation. Participants were very diligent in terms of sending messages to the appropriate entities as well as filling the log of messages. The situation described was communicated along the chain of command and the requisite resources were sent to the location. In effect, the situation was considered to be brought under control, despite a few casualties and injuries due to the incident.

The exercise received a positive feedback. The participants appreciated the realism of the game provided in some key aspects, such as asset exchange and the breakdowns that appeared in the chain of command. Most players commended that the game captured the 'lack of information' situation effectively. This exercise brought out the importance of protocols to be followed during an emergency situation to manage it in an effective manner. While no chief authority was identified before the exercise, in a real-world situation it is imperative that such an authority be defined.

Web Based CPE

A web based CPE was built using the Ruby on Rails framework to overcome issues (message transfer and data analysis) in the paper based exercise. CPE is now deployed on the web and users can participate in exercises remotely. The data can be collected and visualised at the end of the exercise. Administrators can remotely setup exercises, delete/add users etc.

Architecture

CPE is built on the Ruby on Rails (RoR) which is an open source web framework following the Model-View-Controller (MVC) pattern widely used for web based applications. In the MVC pattern the:

- view constitutes what is displayed to the user (user interface)
- model encapsulates the data and provides a data access application programming interface (API)
- and the controller
 - intercept requests from user
 - get data from the model
 - pass data to the concerned view
 - send the view back to the user's browser

The CPE architecture is shown below:

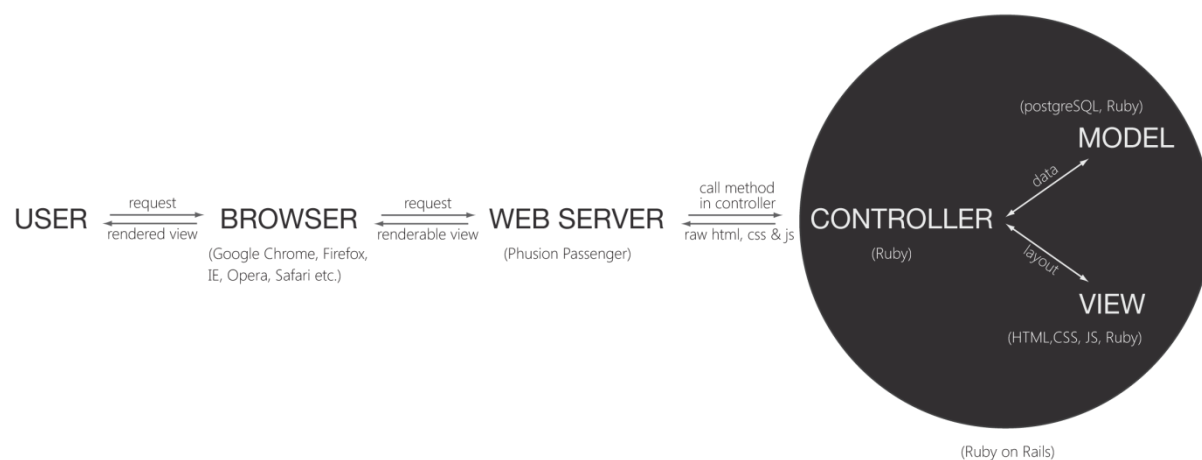


Figure 4.1: CPE Architecture Diagram

CPE runs on the Phusion Passenger application server. Phusion intercepts the HTTP requests from the user's browser and forwards it to the CPE application. The CPE application in turn routes the request based on the URL and its routing table to the concerned controller module in the computer program. The controller module interacts with the model to get application data. It passes this data to the view that is to be sent to the user. It then sends the view along with the data back to the user.

Use Cases

CPE has an administrator and an invited user (i.e. the participant(s)). An invited user can assume the role assigned to him/her in a particular instance of the exercise and participate in that instance. The administrator can perform a variety of advanced actions which include

setting up the exercise, role mapping of the user etc. The cases for invited users and administrators are shown in Figures 4.2 and 4.3.

1. User (Participant)

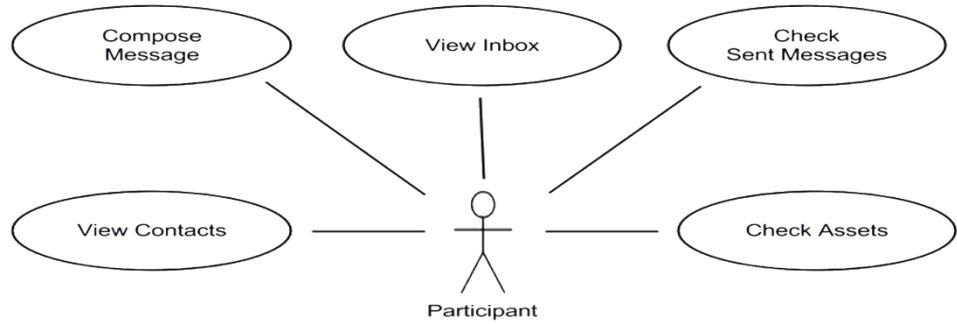


Figure 4.2: User Use Cases

2. Administrator (admin)

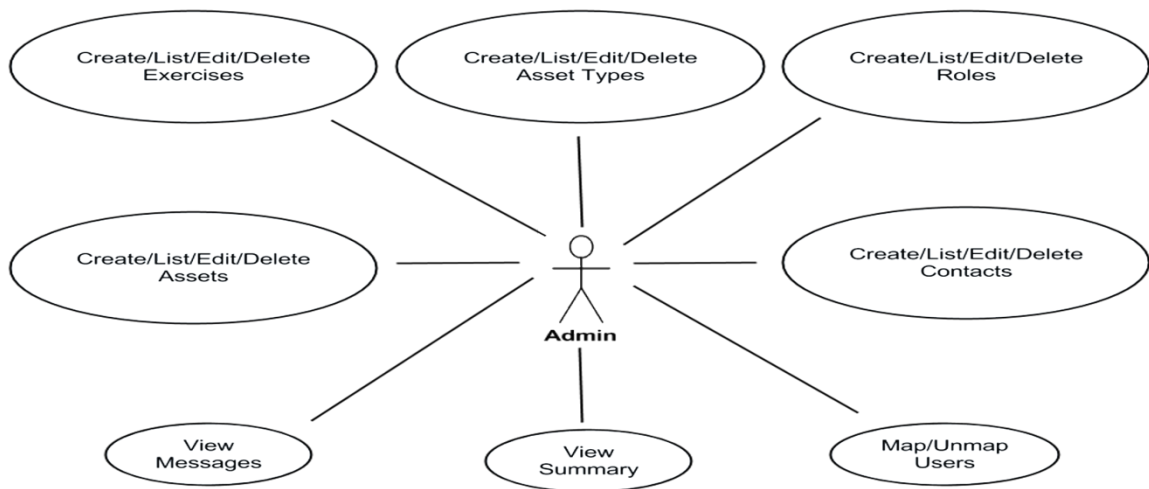


Figure 4.3: Administrator Use Cases

Appendix to Coordination Protocol Exercise

Class diagram

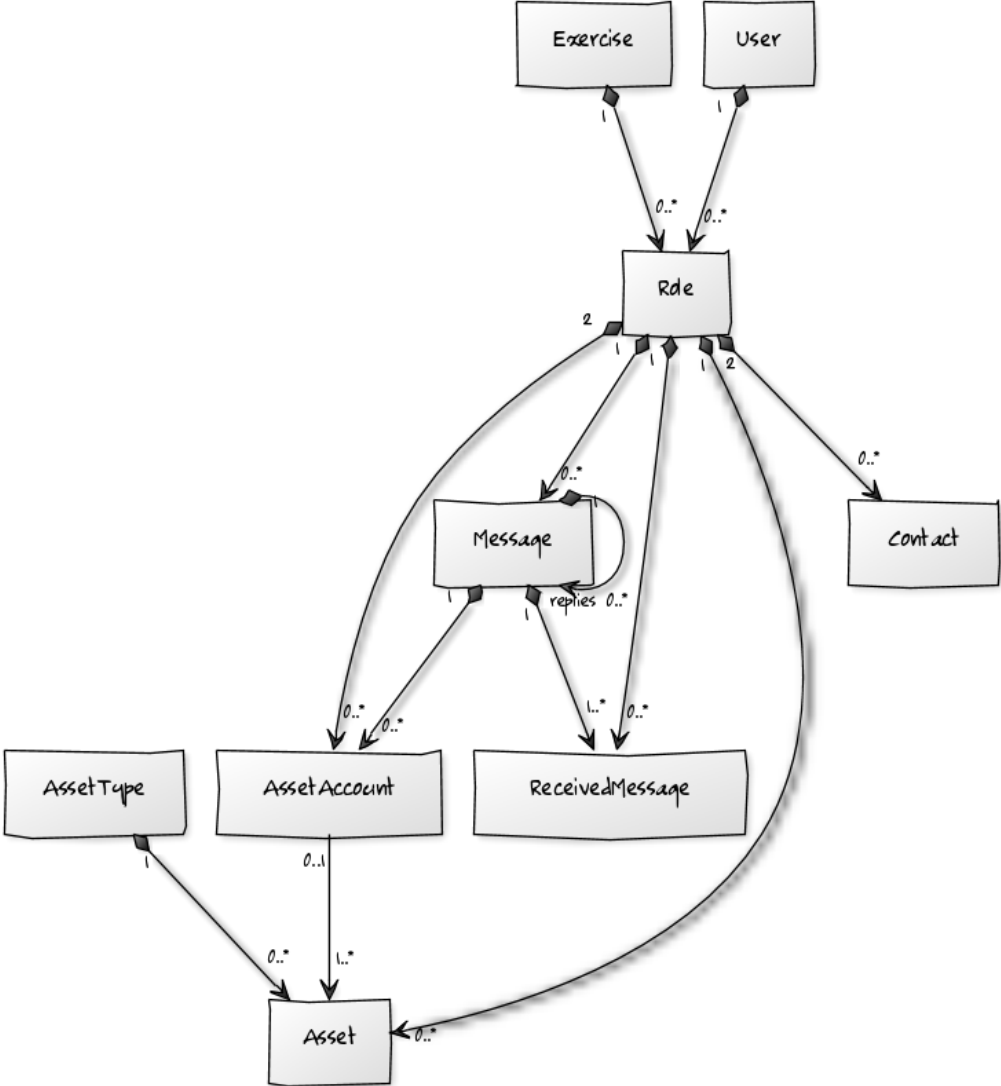


Figure 4.4: CPE Class Diagram

Important Activities and Screenshots

1. Functionality available to user

Send message

A user can send a message to the other users by using the *Compose* screen after login. It displays the list of roles/players in the exercise. A user can send a message to multiple players at the same time. The user can also attach files, deploy assets through message. The screen shows the assets owned and received when the user checks the checkbox *Deploy Asset*. Currently, a user can also deploy multiple assets at a time. But, a user can be assigned only one asset type.

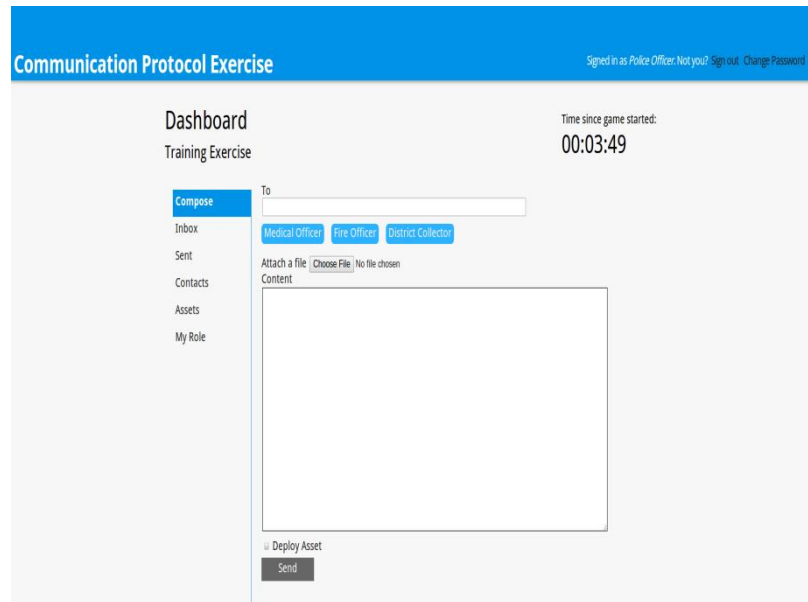


Figure 4.5: Compose Screen

Read Inbox:

The *Inbox* tab shows the list of messages the user has received. This will be updated continuously. Sent time will be shown for each message. If a message has an attachment, it will be shown using hyperlink 'F' which the user can then download. If a message has assets, it will be shown using hyperlink 'A'. The read and unread messages are shown in different colours.

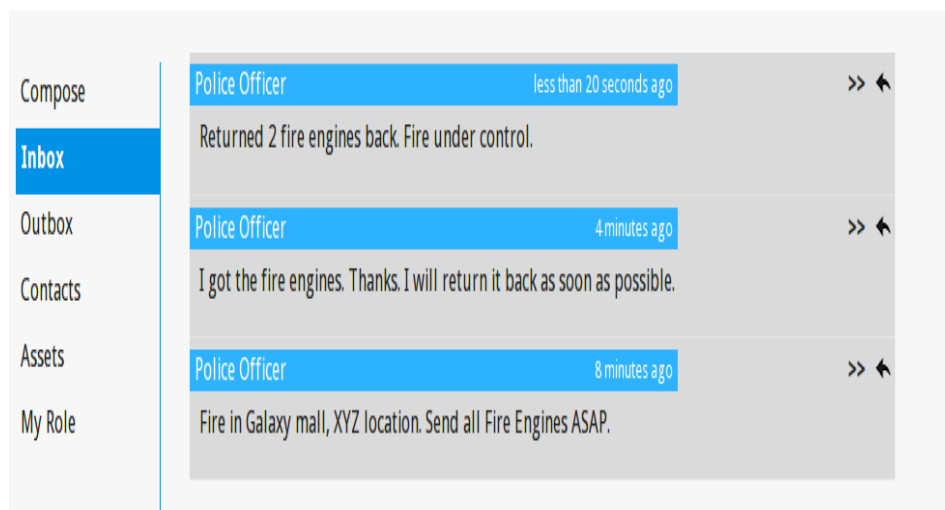


Figure 4.6: Inbox

View Contacts:

The *Contacts* screen shows a list of contacts available for the user. The user will be able to send messages to these contacts only. The contacts list is created by the Administrator.

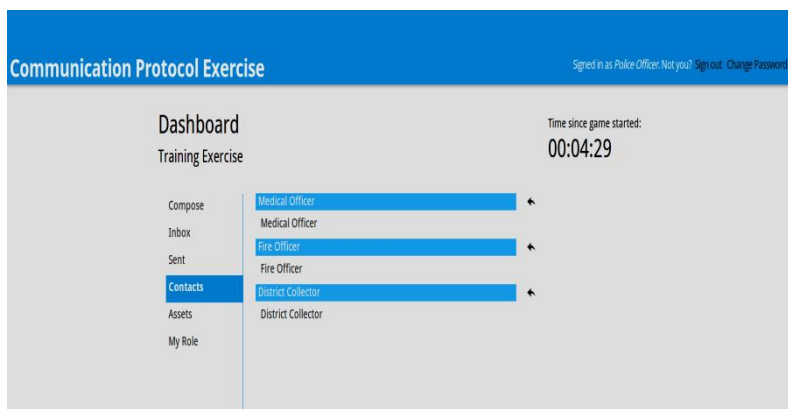


Figure 4.7: View Contacts

View Assets:

The *Assets* screen shows the list of assets owned, deployed and received. The deployed assets list also shows the roles with which each asset deployed and received assets list also shows the roles from which the asset is received.

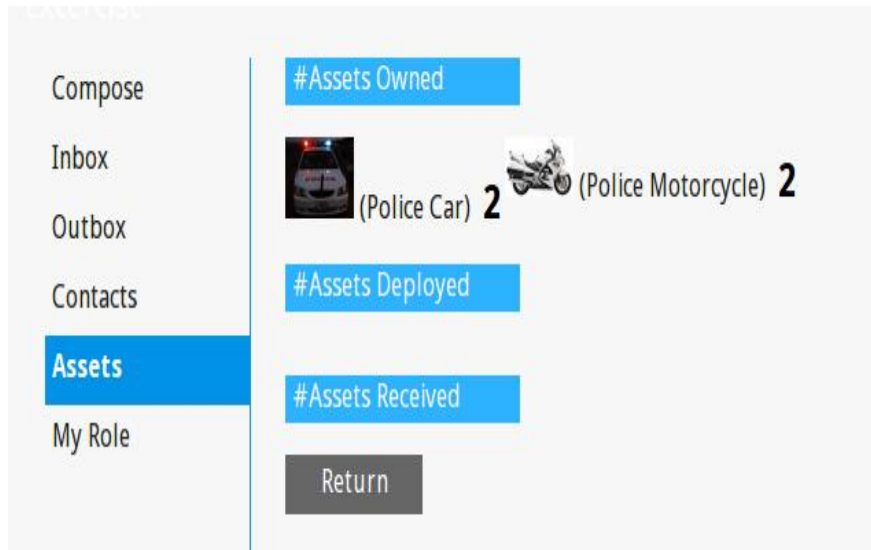


Figure 4.8: View Assets

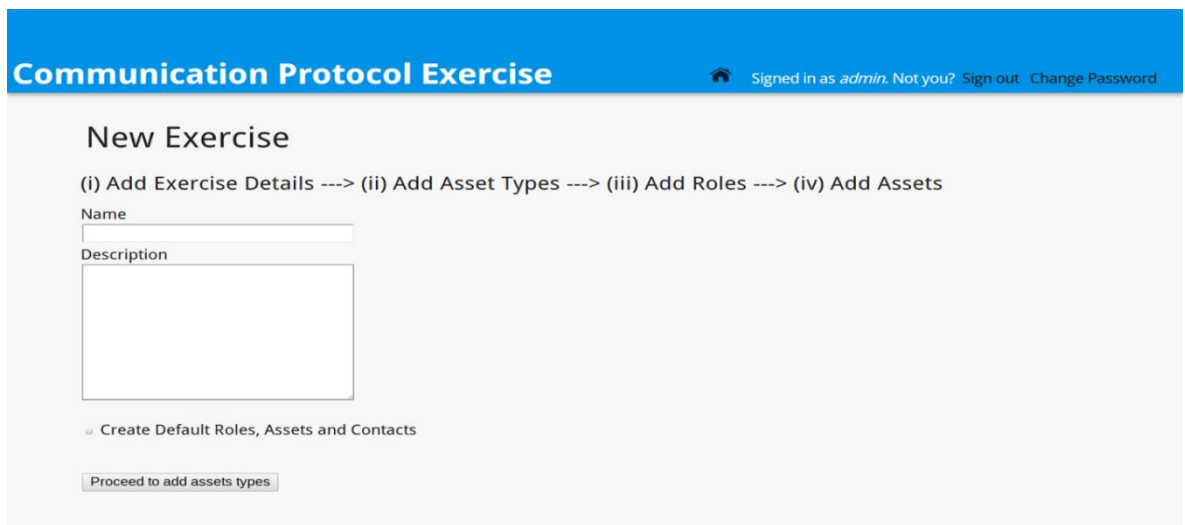
2. Functionality available to Admin

Create Exercise:

Admin can create an *Exercise* by providing the basic details like *Name* and *Description*. The screen which is used to create exercise is shown in Figure 4.9. The administrator can create the exercise by creating asset types, roles, assets and contacts manually. There is also an option to upload an excel file containing information of assets and contacts to save time. This configuration file uses some default roles and asset types in the system. When user checks the checkbox *Create Default Roles, Assets and Contacts*, four default roles - *Police Officer, Medical Officer, Fire Officer* and *District Collector* will be created and these will be assigned to the users - *police_officer, medical_officer, medical_officer, fire_officer* and *district_collector* respectively. Also, there are some default Asset Types in system - *Police Car, Police Motorcycle, Ambulance, Fire Engine* and *Crane*.

Configuration file format:

- File should be of type excel (.xls)
- As shown in Figure 10, the first sheet in the file indicates Assets.
- As shown in Figure 11, the second sheet in file indicates Contacts.



Communication Protocol Exercise Signed in as *admin*. Not you? [Sign out](#) [Change Password](#)

New Exercise

(i) Add Exercise Details ---> (ii) Add Asset Types ---> (iii) Add Roles ---> (iv) Add Assets

Name

Description

Create Default Roles, Assets and Contacts

Figure 4.9: Create Exercise Screen

Role Name	Name of Asset Type	Number of Assets
Police Officer	Police Car	2
Police Officer	Police Motorcycle	2
Medical Officer	Ambulance	3
Fire Officer	Fire Engine	2
District Collector	Crane	1

Figure 4.10: Sample Asset List

Contactator	Contactee
District Collector	Police Officer
District Collector	Medical Officer
District Collector	Fire Officer
Fire Officer	Police Officer
Fire Officer	District collector

Figure 4.11: Sample Contacts Sheet

Create Asset Type

The administrator creates a type of asset by providing details like *Name*, *Capacity* and *Maximum Efficiency*. The admin can also attach an image for an Asset Type.

The screenshot shows a web interface for creating a new asset type. At the top, there is a blue header with the text "Communication Protocol Exercise" and a user status indicator "Signed in as admin. Not you? Sign out Change Password". The main content area is titled "New asset type" and contains the following fields and controls:

- Name:** A text input field.
- Capacity:** A numeric input field with a vertical step-down arrow on the right.
- Maximum efficiency:** A numeric input field with a vertical step-down arrow on the right.
- Image:** A file upload section with a "Choose File" button and the text "No file chosen".
- Buttons:** A "Create Asset type" button and a "Back" button.

Figure 4.12: Create Asset Type

Create Role:

The admin can create different roles for an exercise. This will allow a participant to take on multiple roles in the same exercise. For example, in rural areas, the police chief may also be the fire chief. The admin has to provide details like *Name*, *Description* and *User* for which an appropriate role will be assigned. The user can then participate in the exercise with the assigned role. The role-user combination has to be provided by the admin for each exercise. For instance a role namely *Environment Agent* will be created for every exercise by default and assigned to the default user *environment_agent*.

The screenshot shows a web interface for creating a new role. At the top, there is a blue header with the text "Communication Protocol Exercise" and a user status indicator "Signed in as admin. Not you? Sign out Change Password". The main content area is titled "New role" and contains the following fields and controls:

- Current Exercise:** A label indicating "Current Exercise = Training Exercise".
- Name:** A text input field.
- Description:** A large text area for entering details.
- Select User:** A dropdown menu currently showing "police_officer".
- Buttons:** A "Create Role" button and a "Back" button.

Figure 4.13: Create Role

Map/Un-map Users:

Admin can see list of roles and the users assigned for an exercise. Admin can delete (or un-map) a user from role and can add (or map) another user to the role.



Figure 4.14: Map/Un-map Users

View Message Timeline:

The *Message* timeline shows the communication between roles in the exercise. Admin can view these messages on a screen which will be updated continuously. The timeline shows sender, content and list of receivers, assets and time at which the message was sent.





Sent at	Sender	Receiver	Content	Assets
2012-12-26T10:07:54+05:30	Police Officer	Medical Officer	Returned 2 ambulances. Fire under control.	-
2012-12-26T10:07:13+05:30	Police Officer	Fire Officer	Returned 2 fire engines back. Fire under control.	-
2012-12-26T10:05:56+05:30	Environment Agent	Police Officer	FIRE UNDER CONTROL	-
2012-12-26T10:05:28+05:30	Environment Agent	District Collector	FIRE UNDER CONTROL	-
2012-12-26T10:04:33+05:30	Fire Officer	Police Officer	The 2nd Fire engine is on its way Quote: I got the fire engines. Thanks. I will return it back as soon as possible. Time Sent: half a minute ago -----end of message-----	
2012-12-26T10:03:31+05:30	Police Officer	Fire Officer	I got the fire engines. Thanks. I will return it back as soon as possible.	-
2012-12-26T10:02:57+05:30	Police Officer	Medical Officer	I got the ambulances. Thanks. I will return it back as soon as possible.	-
2012-12-26T10:01:34+05:30	Fire Officer	Police Officer	I am sending 2 fire trucks with 6 fire fighters each. Quote: Fire in Galaxy mall, XYZ location. Send all Fire Engines ASAP. Time Sent: 1 minute ago -----end of message-----	
2012-12-26T09:59:15+05:30	Police Officer	Fire Officer	Fire in Galaxy mall, XYZ location. Send all Fire Engines ASAP.	-
2012-12-26T09:58:31+05:30	Medical Officer	Police Officer	I am sending two Ambulances	 
2012-12-26T09:56:53+05:30	Police Officer	Medical Officer	Fire in Galaxy mall, XYZ location. Send all ambulances ASAP.	-
2012-12-26T09:53:02+05:30	Environment Agent	Police Officer	FIRE AT GALAXY MALL. Few hundred people trapped inside.	-

Figure 4.15: View Message Timeline

View Summary:

Admin can view the summary of the exercise after it is completed. It provides the basic information like number of messages and number of times assets are deployed. It also shows how a particular role performed in the exercise by displaying the percentage of messages sent by that role and the percentage of messages sent during a time frame.

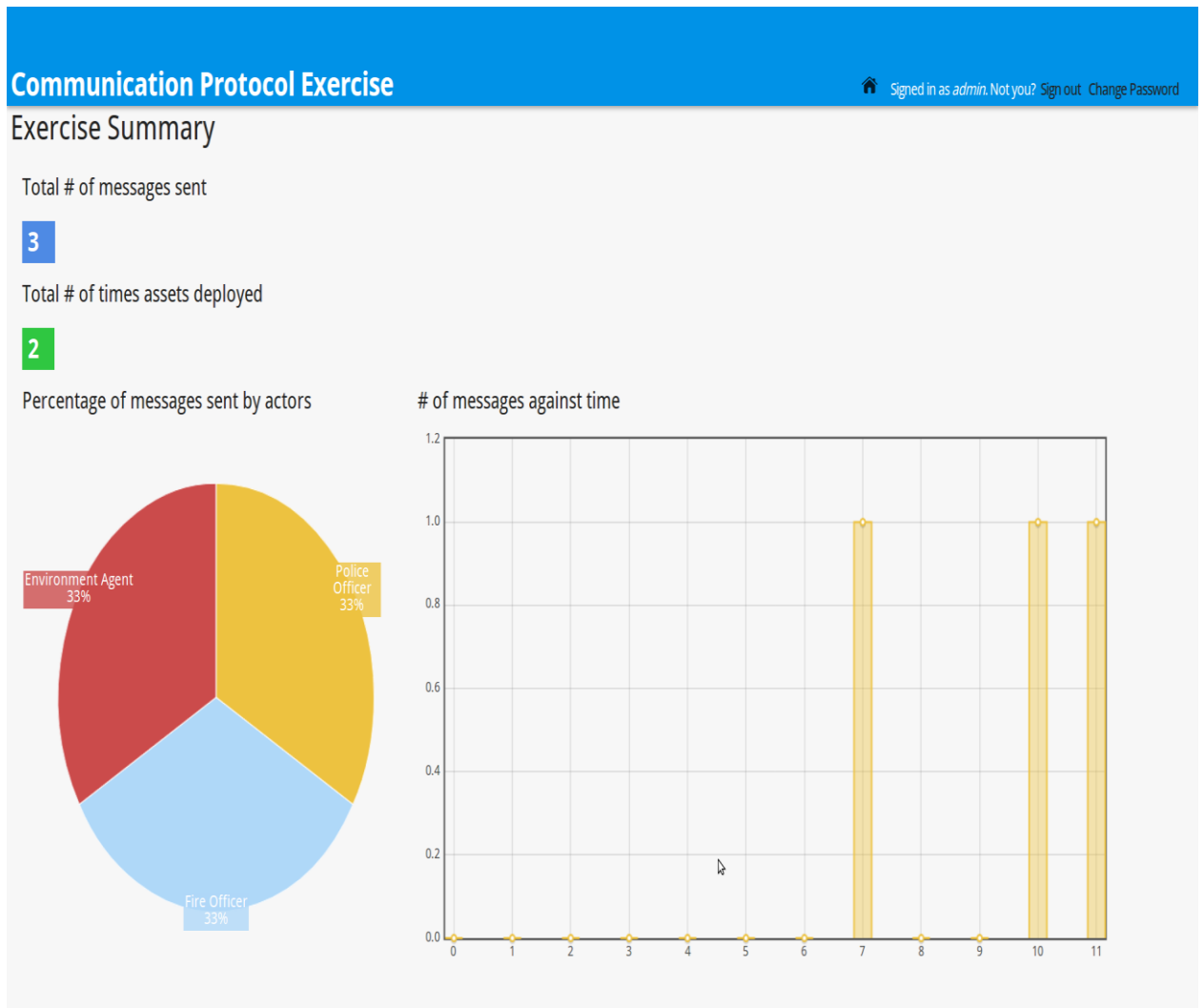


Figure 4.16: View Summary

Observations and Future Enhancements

The communication protocol exercise gives us valuable insight on disaster management protocols and communication patterns amongst emergency respondents. Some of them are listed below:

- Hierarchical issues within the disaster management team
- Data liberation policies to make certain data available in case of an emergency
- Testing correctness of a protocol
- Modification to protocols to keep them relevant and up to date

There were concerns regarding the institutional framework followed in the exercise as it did not include one-to-one correspondence with the existing real-world framework. In a real-world situation, roles are assumed by one person. Another important observation was that in case of some disasters, people with defined roles meet at physical sites to discuss protocol.

Taking into account some of these concerns, CSTEP designed CPE as a framework instead of an instance of a particular exercise. This enabled the spawning of multiple exercises, each with its own communication hierarchy, asset base and role definitions. This *configurability* makes the CPE framework very user-friendly.

5. First Responder Game

Introduction

First responders in an emergency face several difficult situations. These may include blocked access routes, physical injuries to people and property destruction. In this state, it may be overwhelming for an individual to focus on the core function. This situation is further exacerbated if the individual does not know about the assets available. We have developed a game/training tool called the “First Responder Game” that simulates a disaster and allows the first responder to visualise a response. This also allows a trainer to find out if the response is appropriate and adequate or if there is scope for improvement.

Game Prototype

The prototype testing is based on a bomb scare scenario with a mission to help police lay barriers and cordon off a street. The scenario is at a popular mall in Bangalore. The main objective assigned to the first responder (or player) is to block oncoming traffic using barricades. After laying out the scenario, a list of assets along with a set of other possible “actors” (police, fire department, bomb squad) is prepared and placed in a framework. This framework helps to realise the interaction between the actors and the assets.

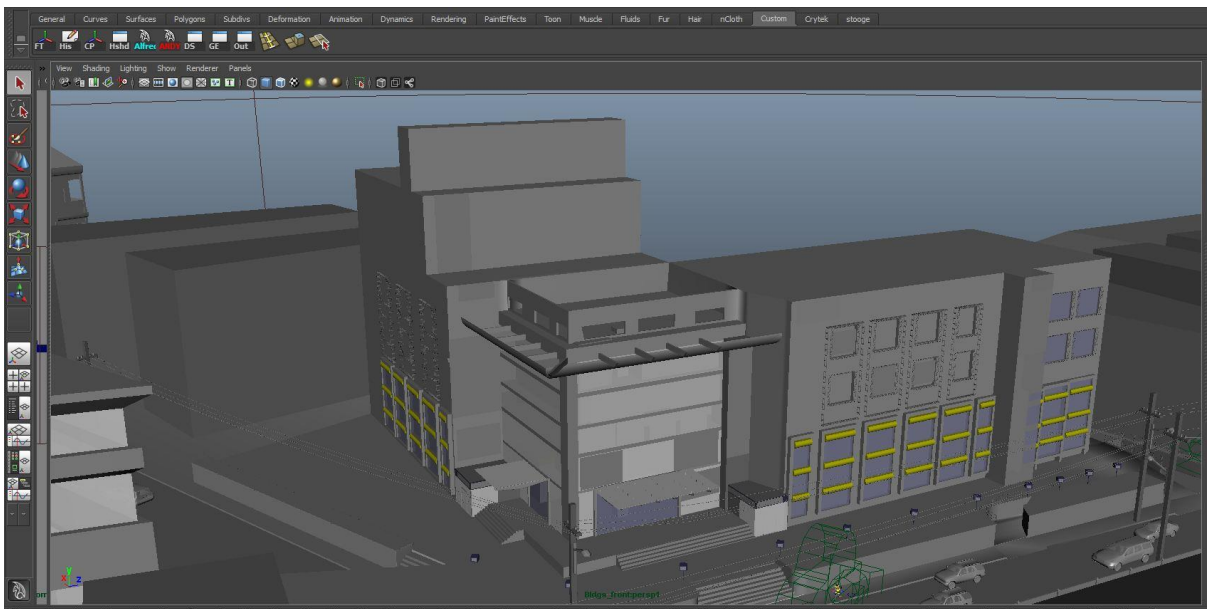


Figure 5.1: Rendering of a Mall Structure (Image from Maya Viewport)

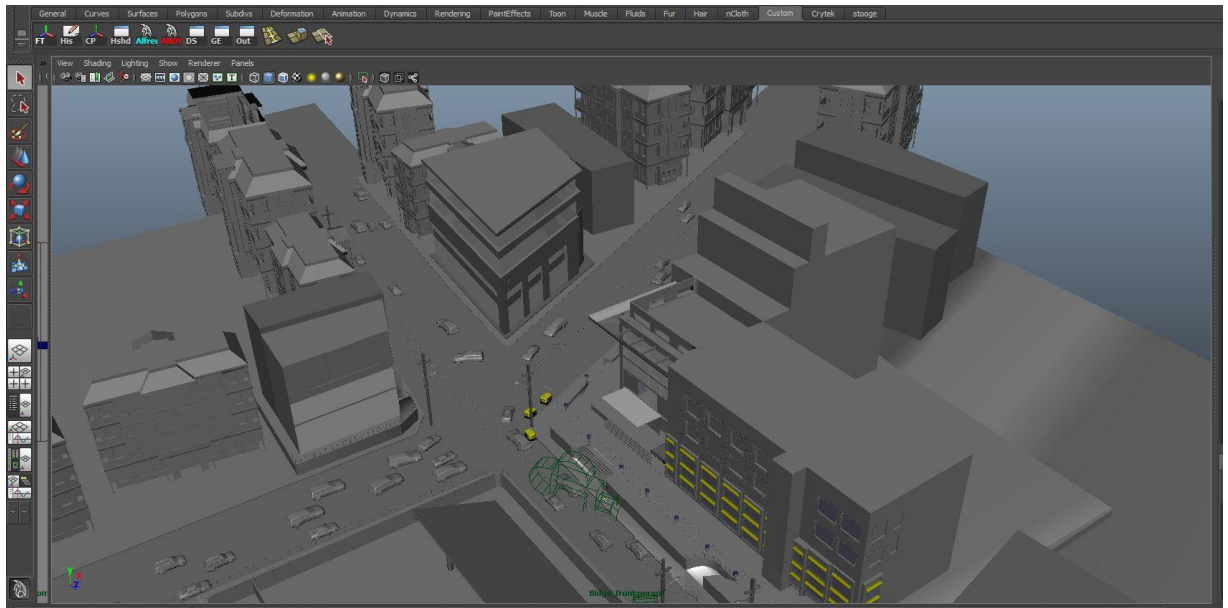


Figure 5.2: Rendering of a Mall and Surrounding Areas (Image from Maya Viewport)

Game Development Process

The game was developed using CryEngine3. It allows for real-time rendering of scenarios, which aids in speeding up the development process. In addition, the next-generation graphics engine of CryEngine3 allows us to develop realistic multiplayer game scenarios. These allow us to integrate multiple actors into a single scenario with every actor having a set of roles. Engaging participants in a space which is relatively unknown, gives an opportunity to adapt, improvise and interact with each other, leading to strategic learning and real-time decision making. Thus, the inclusion of games in 3D space and the multiplayer aspect may lead to a better understanding of the short comings in communication and decision making.

The steps followed were:

1. Creating a mall and some of the surrounding areas.
2. 'Texturing' (a process to make the visuals look more realistic) the mall and other buildings in close proximity to the mall.
3. Depicting the scene (called 'renders') and creating walk-through animations, based on 2 D drawings in the main library. The 3D models with their respective textures were then optimised for importing into CryEngine3.
4. Importing and positioning individual pieces then in CryEngine3 initiating the level design process. This included set design, asset placement and populating the scene with trees and vegetation. To do this, one must have photographs/videos of the location ahead of time. CryEngine3 allows modification of material properties based on the requirement of the developer. We had to setup physics proxies based on building materials. Physics involved setting up of mass, density, materials that

shatter etc. Once level design was complete, logic and mission setup was initiated. For the first responder mission, we used the flow graph logic within CryEngine3.

5. Actors were setup at required places and a spawn point for the first responder was setup. (A spawn point is used for placing a player at a desired location at the start of a mission). The player gets on screen objectives about the mission which gets updated. . Upon mission completion, the player can decide either to restart the mission or move on to the next mission/level.

Screenshots

The following screenshots explain the process, including playing the game (for the situation described earlier in this chapter), in detail.



Figure 5.3: The Layout (Image from Maya Viewport)

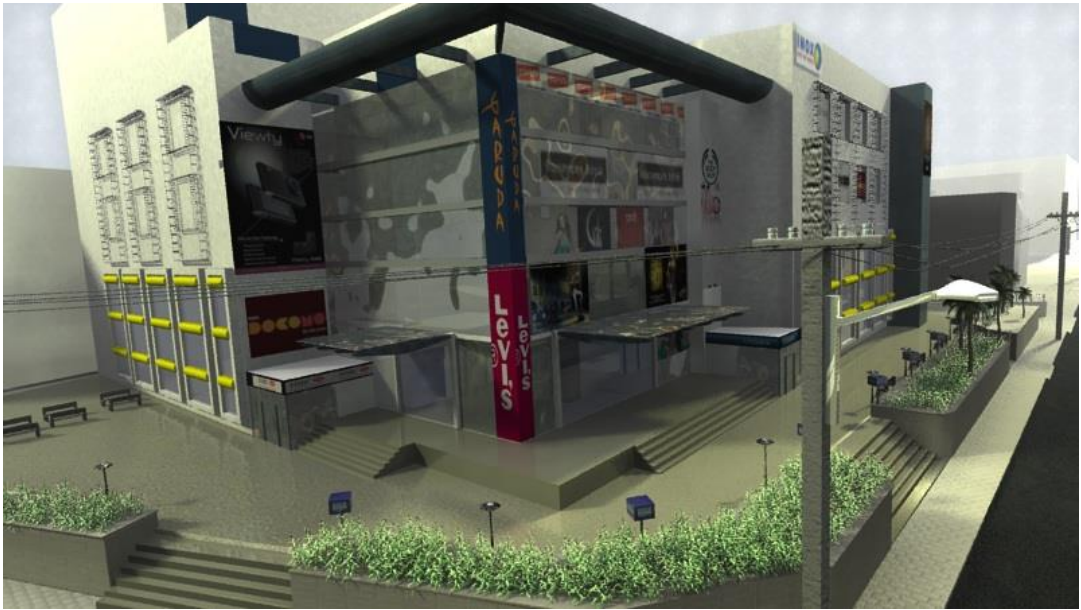


Figure 5.4: The Mall from a Different Angle (Image from Maya Viewport)



Figure 5.5: Another View of the Mall and Side Streets (Image from Maya Viewport)



Figure 5.6: Design Imported into CryEngine3 (for Placing Vegetation, Assets, Trees etc.)

Notice (in the figure above) the heightened level of detail as compared to the previous image that show the same view.



Figure 5.7: A Side Street in CryEngine3 (Level Design)



Figure 5.8: The Side Street from another Angle

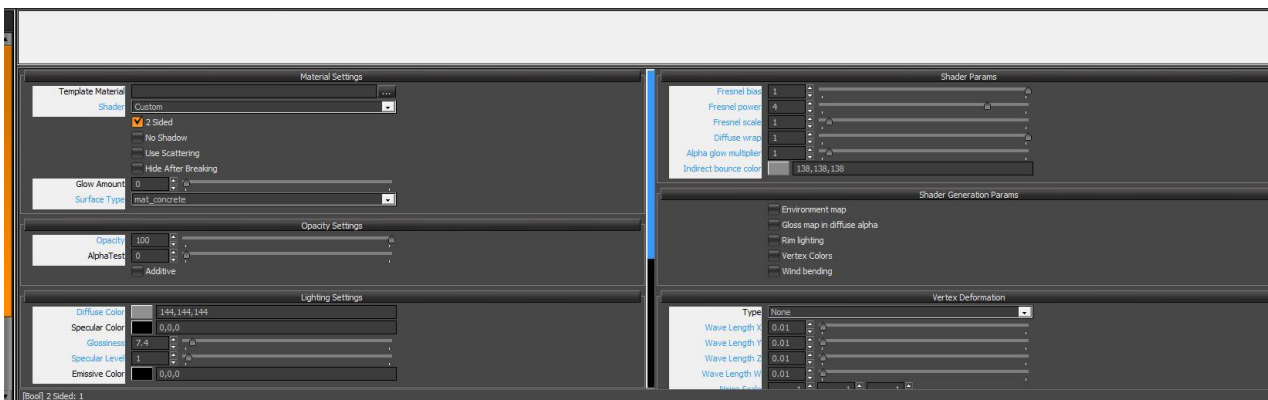


Figure 5.9: : CryEngine3 Material Editor



Figure 5.10: Actors Introduced via CryEngine3



Figure 5.11: Actor Performs and Finishes Actions

6. 2-D to 3-D Conversion and Comparison of Game Engines for Simulation

Introduction

3D walkthroughs of buildings and locations help emergency agencies have better situational awareness prior to entering an area struck by a disaster. In scenarios such as a fire engulfing a building, a hostage rescue situation or unstable structures, this walkthrough can be of great help in providing prior information and awareness to the appropriate agencies. This also helps locate assets and can give a real-time view as the generated model can be updated dynamically. We have converted Computer Aided Design (CAD) drawings of building plans to 3D models using modelling tools and then deployed these models in CryENGINE3 (a game engine for real-time rendering). To convert 2D plans into 3D models, a variety of tools are required. Some of these are:

Autodesk Maya

Autodesk Maya is the state of the art 3D computer graphics software to create interactive 3D applications, including video games, animated films, TV series and visual effects. Maya provides powerful integrated animation, modelling, simulation, rendering, match-moving and compositing tools on a robust and extensible Computer Graphics (CG) pipeline core. It has next-generation display technology, accelerated modelling workflows and new tools for handling complex data.

Game Engines

The models we created are rendered in real time using game engines such as *Unity*, *Source Engine*, *CryENGINE3®*, *Unreal Development Kit (UDK)* etc. However we chose CryENGINE3 and UDK as our preferred platforms because of their vast code libraries, engine support and large developer base.

1. CryENGINE3

CryENGINE 3, developed by Crytek GmbH, is an innovative solution in rendering high quality 3D elements without significant performance impact or compromise on graphical quality. This is across all supported platforms and is 100% real-time. Developers can work with tools and technologies that allow the creation of games in 3D, regardless of the display technology utilised to deliver the most immersive games ever created.

2. UDK

UDK is the free version of the award-winning Unreal® Engine 3, a software development framework used to create computer and video games, 3D simulations, TV shows, films etc.

Software Requirements

The following software configuration has been used in this tool:

- Operating System: Windows 7
- Autodesk Maya 2012 onwards
- CryENGINE3 Resource Compiler for Maya 2012/13
- FBX Maya2012/13 Exporter for UDK
- CryENGINE3 SDK or Unreal Development Kit

Assumptions

Some assumptions made are as follows:

- Availability of building plans in computerised (png, tif or dwg) format, usually obtained from a civic agency through a centralised database (where soft copies of all building plans can be uploaded). In terms of policy, this has to be made mandatory after the approval of a plan. If deviations are made to the approved plan, such deviations are also to be made available.
- Availability of photographs (uploaded after construction has been completed).

Detailed Modelling Process - 3D Environment

The process for 3D modelling comprises of:

1. Optimising and Importing Building plans

The building plans should have the following data embedded

- Wall thickness (maybe using line width) and height
- Construction materials to be used in an ideal situation (for example hollow bricks, concrete, glass)
- Electrical cable layouts and water and any other pipelines
- Doors, windows, AC ducts, vents and emergency exits

The building plan can be imported in the top view of Maya. This will generate a background image on which the modelling can be done.

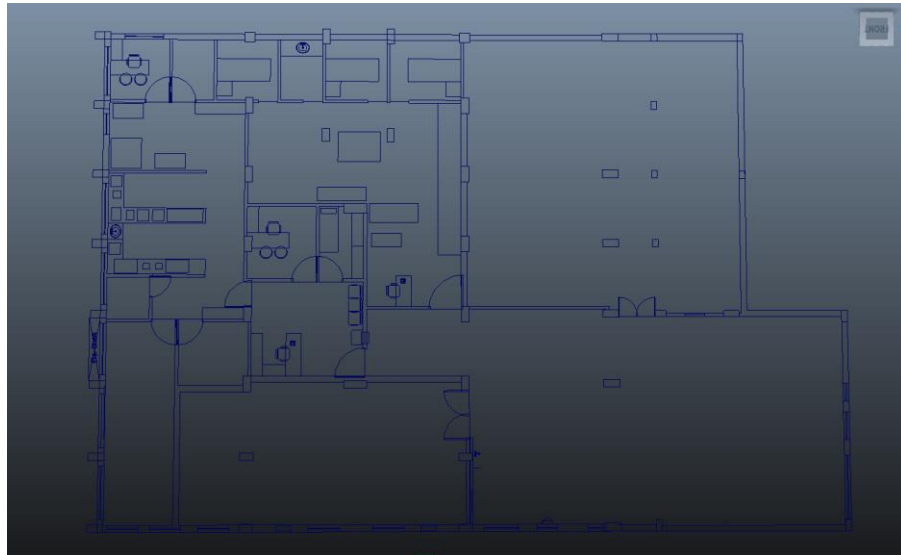


Figure 6.1: Sample Floor Plan

3D Modeling and Development Process

The modelling process begins with extruding the exterior walls and moving towards the interior (Figure 6.2). Once the walls have been extruded, doors and windows are placed according to the building plan.

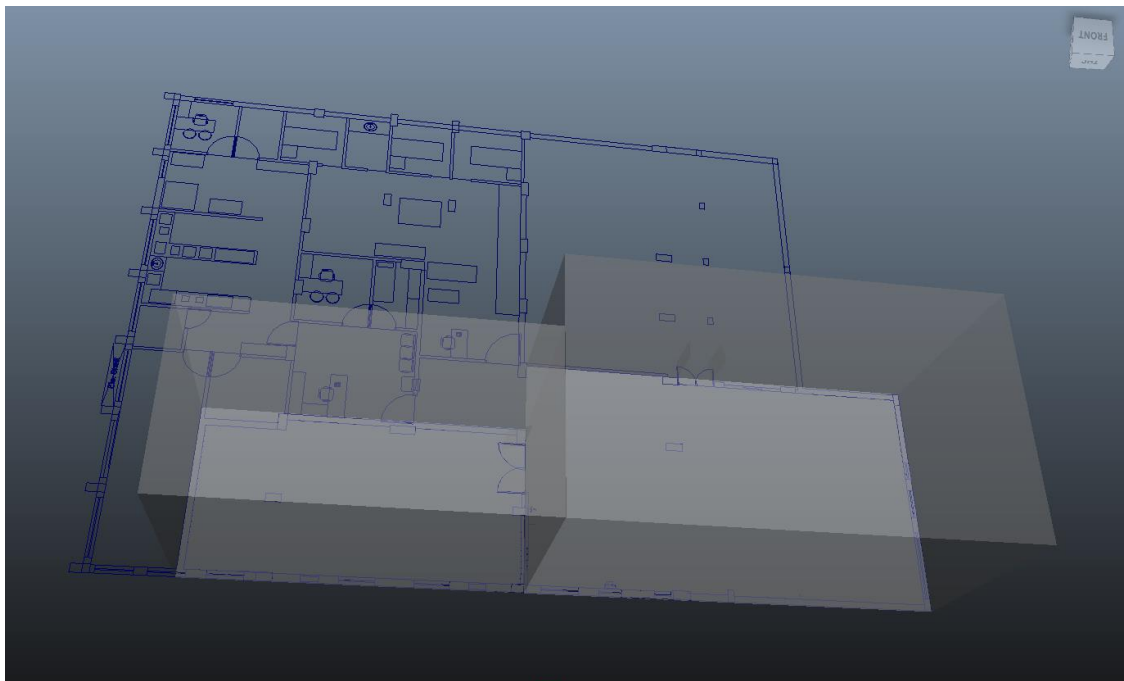


Figure 6.2: Sample Building Blocks

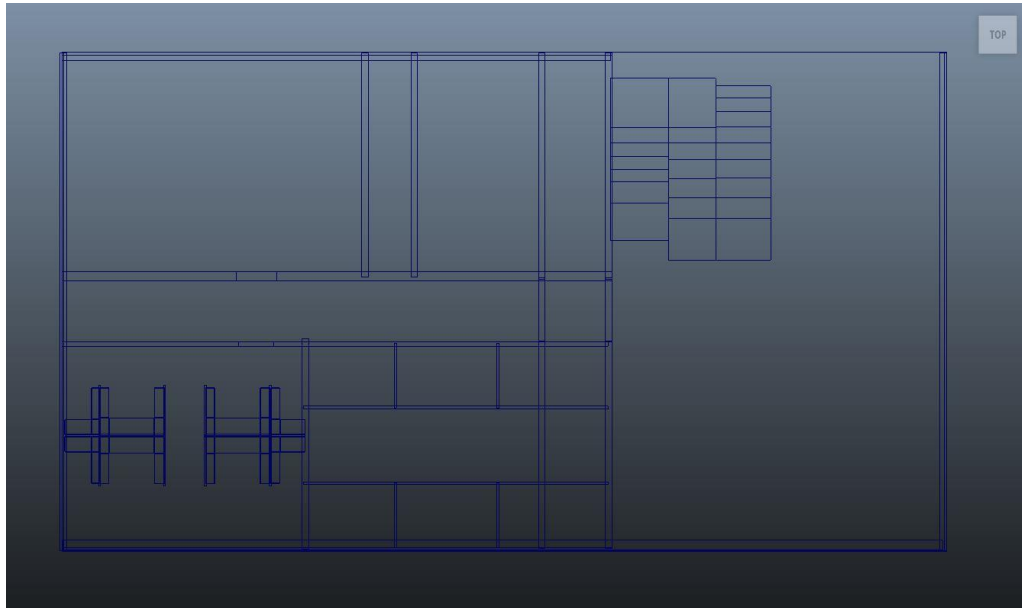


Figure 6.3: Floor Plan of a Private Building in Bangalore

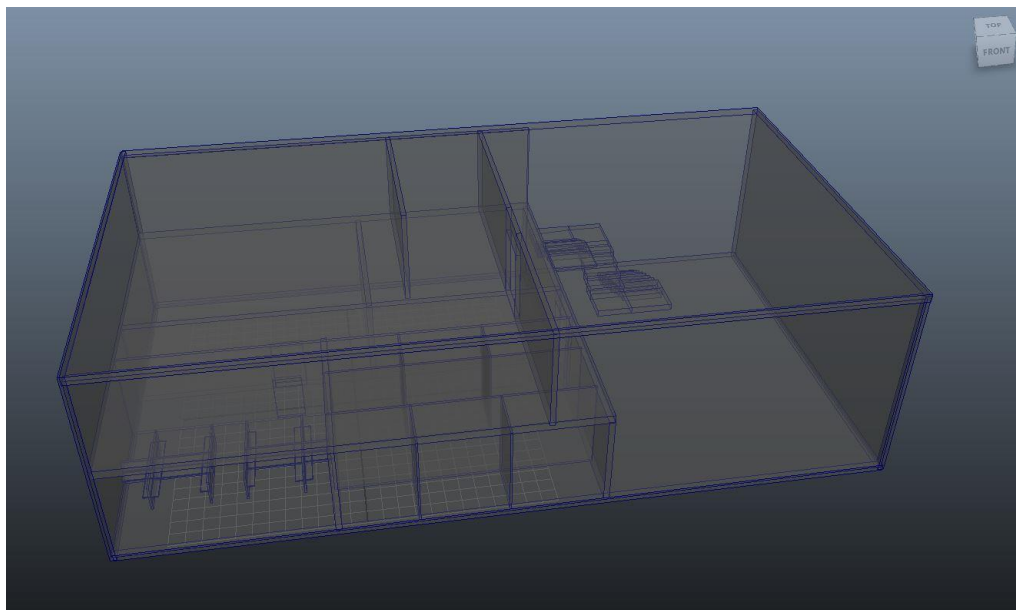


Figure 6.4: Building Block of a Private Building in Bangalore

Polygon Optimisation of Models for CryENGINE3 and UDK

Polygons are used in computer *graphics* to compose images that are three-dimensional in appearance. In other words, any structure can be represented by a limited number of polygons. Therefore, optimising the polygon count in models is important to keep the number of poly faces as low as possible without losing detail. A lower polygon count will lead to fewer

loads on the hardware. This optimisation helps deploy the model easily on devices with lower processing power.

Texturing

Texturing adds detail to the 3D models by providing surface textures and implementing material properties (glossy, matte, surface irregularities etc). Texturing provides realism to the models and also helps with simulation using the material properties of the texture.

Optimising and exporting the 3D models for appropriate game engines

The models have to be prepared in Maya based on the type of game engine used in the pipeline. For CryENGINE3, the models have to be placed in a single group and naming conventions have to be followed. For example: If the model is named Block1, then it should be under a group called "block1_group" and this group should be under another group named "cryexportnode_block1". The CryENGINE3 resource compiler should then be used to export the models in ".cgf" format. For UDK, the models can be exported using the Maya FBX exporter. Under the export node, the triangulate option should be highlighted before export.

Game Development Process

The game development process requires optimising the models and then performing a level design. Following are the steps:

1. Optimising and Importing 3D Models

Maya models have to be optimised with respect to polygon count, triangulation (Maya automates this process) and edge consistency.

1.1. CryENGINE3

The CryENGINE3 Resource Compiler optimises the models and prepares them to be imported into CryENGINE3. The models (.cgf files) should be placed inside the game.pak file which is located in the installed directory.

1.2. UDK

BX exporter in Maya renders the models in ".fbx" extension format. The file can then be imported into UDK directly using UDK's import function.

2. Level Design and Setup of Models

Level design is the process of designing and creating an environment and the locales in which the models will be imported.

2.1. CryENGINE3

The appropriate terrain is setup by either using a terrain modelled in Maya or using CryENGINE's Terrain Generator. Once the base is setup, the other models that need to be imported are placed accordingly.

2.2. UDK

Just as done in CryENGINE3, the terrain in UDK can either be modelled or generated using UDK's Terrain Mapping tool. Once the base is setup, the other models that need to be imported are placed accordingly.

3. Texturing and Detailing Models

As explained before, texturing is the process where the materials are applied to the building. The details include smoothness and certain properties like colour, varying depth etc.

3.1. CryENGINE3

CryENGINE3 uses the "Material Editor" tool as its texturing unit. The materials, textures and surface properties of models are assigned using the material editor.

3.2. UDK

Materials in UDK are assigned using the "Content Browser". The material properties can be edited using the material editor in the content browser.

4. Lighting Setup

Lighting is done in the game engine to set the time of day. Good lighting influences the realism of the level. Every effort is made to duplicate the lighting as exists in a similar, real-world setting.

4.1. CryENGINE3

CryENGINE3 features near-reality natural lighting using global illumination. It is capable of creating soft shadows that dynamically respond to natural movements in real time. High resolution, perspective-correct and volumetric smooth-shadow implementations are also included in CryENGINE3. This works with both static and dynamic objects.

4.2. UDK

Unreal Engine uses ambient occlusion, per-pixel lighting, fill lighting and fully dynamic specular lighting and reflections for lighting its scene. Since the 'Global Illumination' in UDK is not as powerful as the one in CryENGINE3, spot lights, point lights, fill lights are used as a substitute.



Figure 6.5: An Imported Model in CryENGINE3

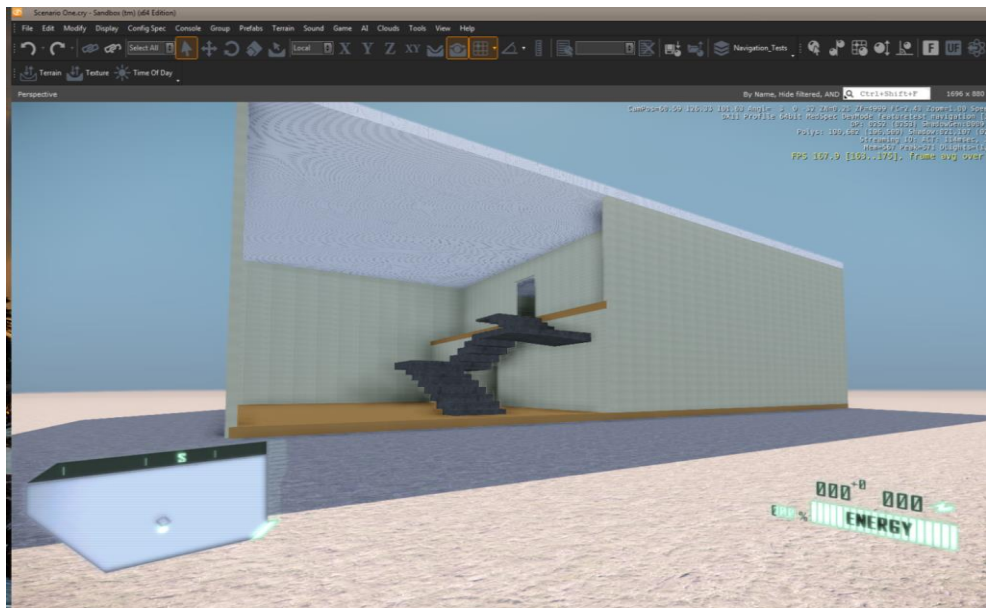


Figure 6.6: Front View of Interiors with CryENGINE3

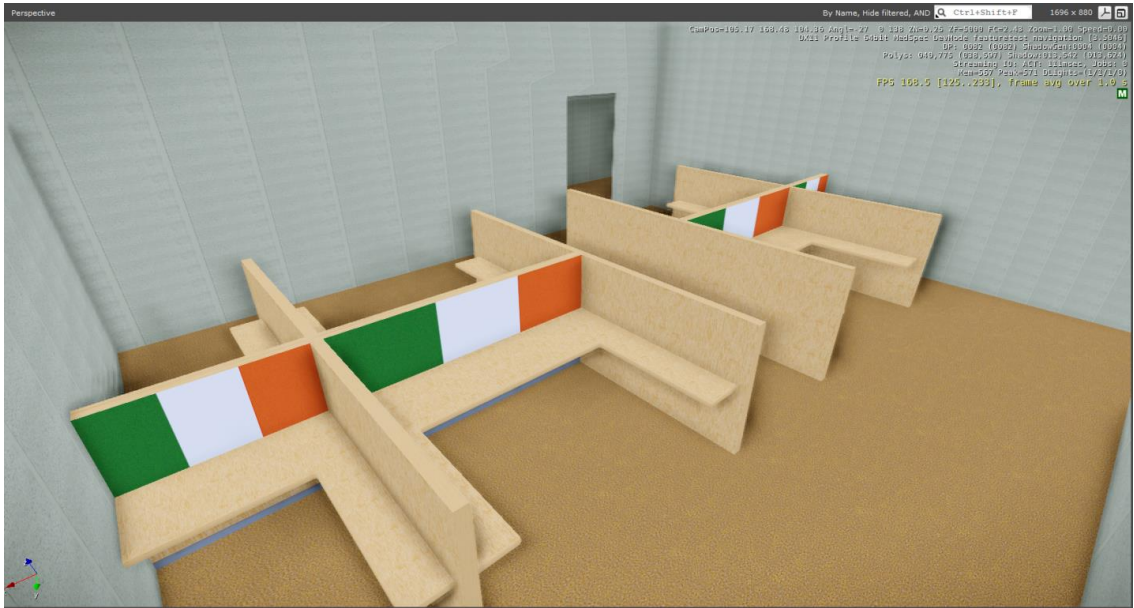


Figure 6.7: Top View of Interiors in CryENGINE3

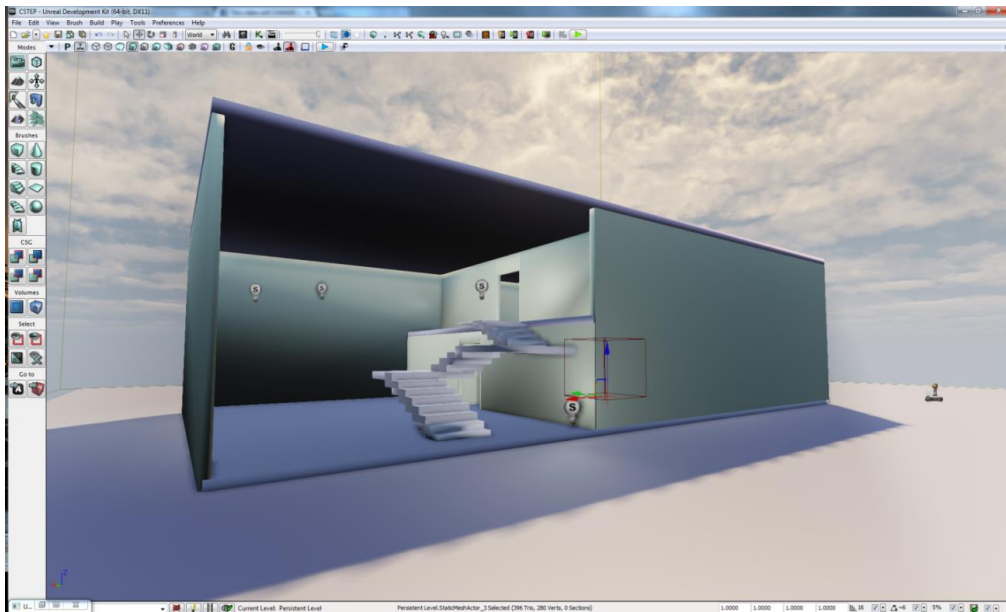


Figure 6.8: An Imported Model in UDK

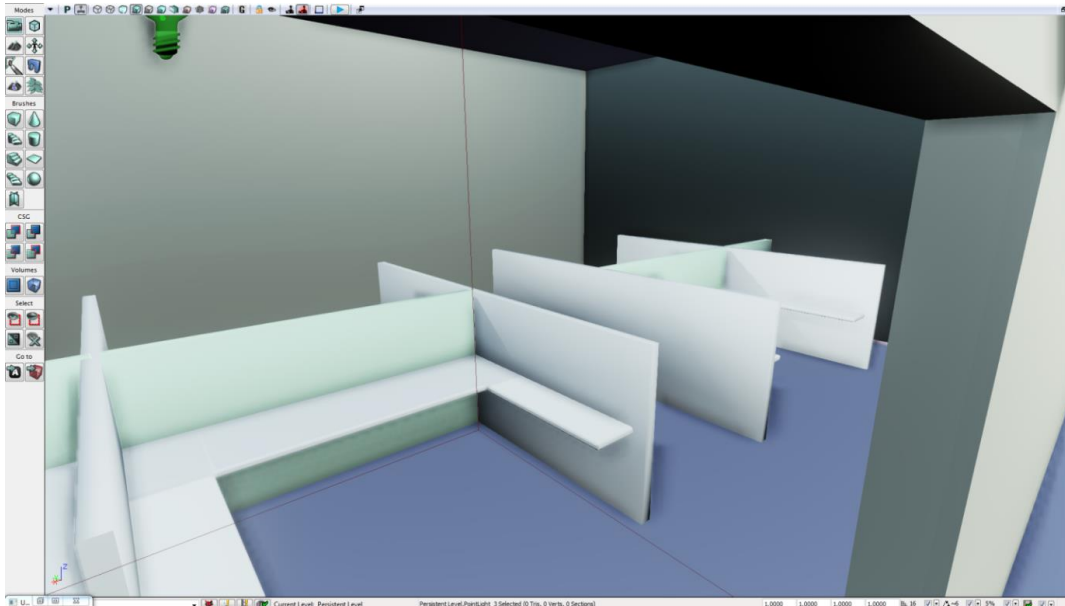


Figure 6.9: Interiors in UDK

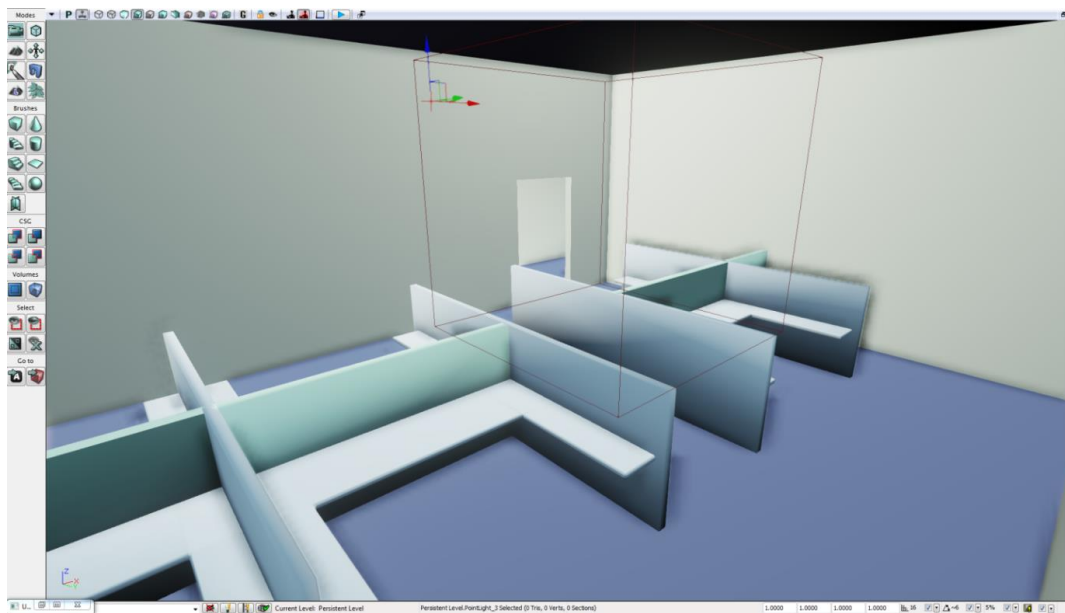


Figure 6.10: Interiors in UDK (with more detail)

5. Assigning Spawn Points for Start of Walkthrough

Spawn points define the positioning of the player before the walkthrough begins.

5.1. CryENGINE3

5.2. Player spawn points need to be assigned using the spawn point tool at appropriate location after the models have been setup. This avoids delay in the movement at the beginning of the walkthrough.

5.3. UDK

Player spawn points need to be assigned using the spawn point tool at appropriate location after the models have been setup. This avoids delay in movement at the beginning of the walkthrough.

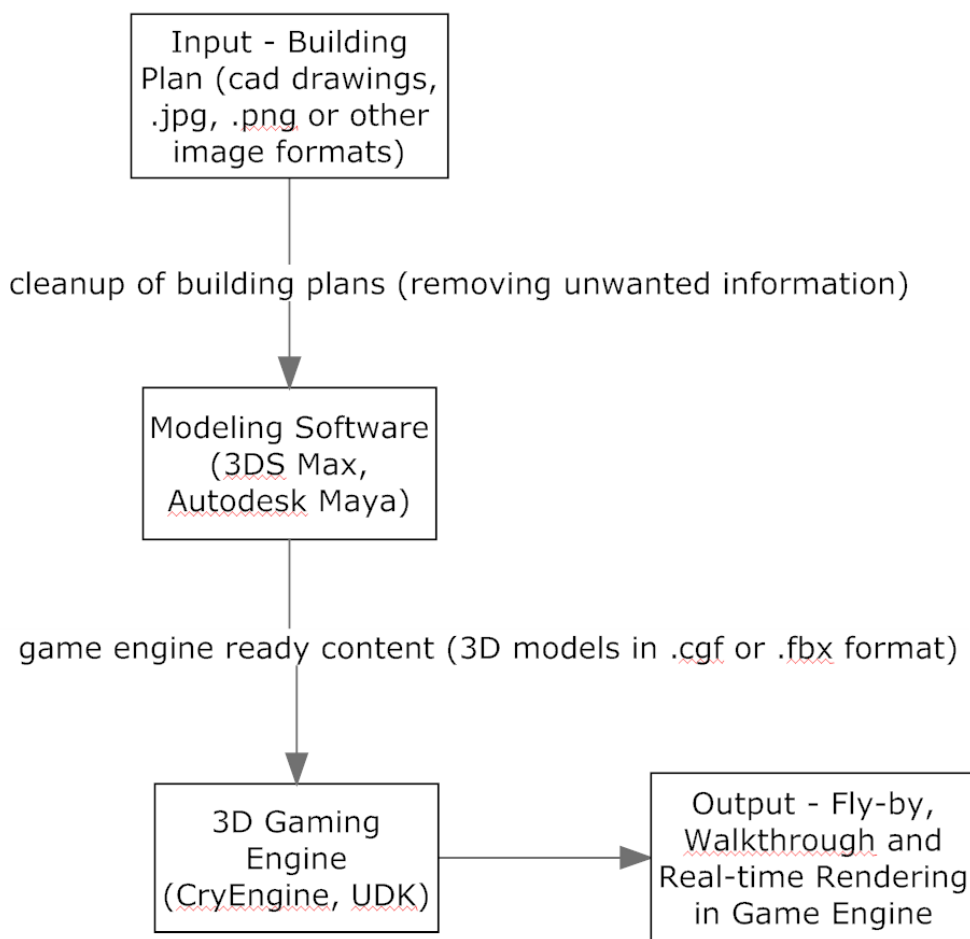


Figure 6.11: Flow diagram for Game Design

A Comparison of CryENGINE3 and UDK

Table 6.1: Comparison of UDK and CryENGINE3

	Name	CryEngine 3	Unreal Development Kit (UDK)	Autodesk Maya 2013
Platforms	Windows	Yes	Yes	
	Mac	No	No	
	Android	No	No	
	iOS	No	Yes	
	C++ Interface	Yes	No (Unreal Script)	
API's	Graphics API	DX 9, 10, 11	DX 9, 11	
	Physics Engine	Custom	PhysX	
Game Features	Path Building	Yes	Yes	
	Unlimited Sized Worlds	Yes	No	Not a realtime rendering engine
	World Editor	Yes	Yes	
Collaborative World Editing	No	No		
Editors	GUI editor	No	Yes	
	Deferred Renderer	Yes	Yes	
	Forward Renderer	No	No	
	Ambient Occlusion	Yes	Yes	
Graphics Features	Parallax Mapping	Yes	Yes	
	Per Object Motion Blur	Yes	Yes	
	Physical Clothing	No	No	
Physics Features	Ragdolls	Yes	Yes	
	Destructible Objects	Yes	Yes	
PRICE		Limited: Free or USD 1,000,000?	Limited: Free or USD 99 + 25% Royalties	USD 3500

Skills Required

- 3D - Modelling, Texturing, Lighting
- Game Development

Costs

- Autodesk Maya - Rs. 160,000 (approx)
- CryENGINE3 - Limited Capability Edition is Free. Commercial version costs \$1.2 million to license
- UDK - Limited Capability Edition is Free. Commercial version costs \$99 + 25% Royalties

Conclusion

The use of game engine technologies for converting 2D plans to 3D models within a limited time can provide valuable situational awareness to emergency agencies. Using the state of the art technology available today, the time required in completing 2D to 3D walkthrough of reasonable accuracy and quality averages around 11 minutes, a fact that might be critical in emergency scenarios. However, it must be ensured (perhaps via a policy definition) that plans and blueprints of all important buildings are available in the computer. The entire task is skill-intensive and needs manual input. Automation of certain procedures needs to be experimented which can further reduce skill dependency and time required to convert each walkthrough cycle. Further work on game engine technology can provide additional insight and shortcomings in current policies and procedures.

7. Triage Exercise

Introduction

The loss of life and limb during emergency incidents such as a fire or an accident can be mitigated by proper and prompt medical attention. Though the first responders (security services, police, firemen) are trained (or should be trained) in basic medical procedures, the hysteria and confusion during incidents, the volume of casualties and the lack of paramedical facilities may lead to poor allocation of resources, inappropriate treatment and unacceptable results. A tool that helps first responders to optimally classify victims so as to minimise exacerbation is needed to address this problem.

Triage is a quick process of determining the need for priority treatment for a victim based on the severity of his/her condition. We have developed a tool based on the START (*Simple Triage And Rapid Treatment*) model that provides various computer-based exercises to train the first responders. These exercises comprise of several steps in identifying a victim's state and tagging him/her based on the criticality of the condition.

The Triage Exercise Tool

The responder being trained first gets a textual description of the chosen casualty, along with an image. The values for Respiration, Pulse and Mental status (RPM) are also provided [*Note: In a real-life situation, the responder would to determine these values himself/herself, usually as the first step*]. Next, the trainee analyses and 'tags' the casualty (using the appropriate button on the screen) based on the START procedure. The tags are coloured red, yellow, green and black. The performance of the first responder can be evaluated based on the accuracy of tagging and the time taken to tag each casualty. In addition, the tool has a live polling functionality that (a) allows one to evaluate trainee performance in real time so as to provide instantaneous feedback, and (b) compare the performances of one team against another. More details about the tool are provided in subsequent sections.

The S.T.A.R.T. Method

The START TRIAGE flowchart is shown below. The colours depict the appropriate tags for the victim:

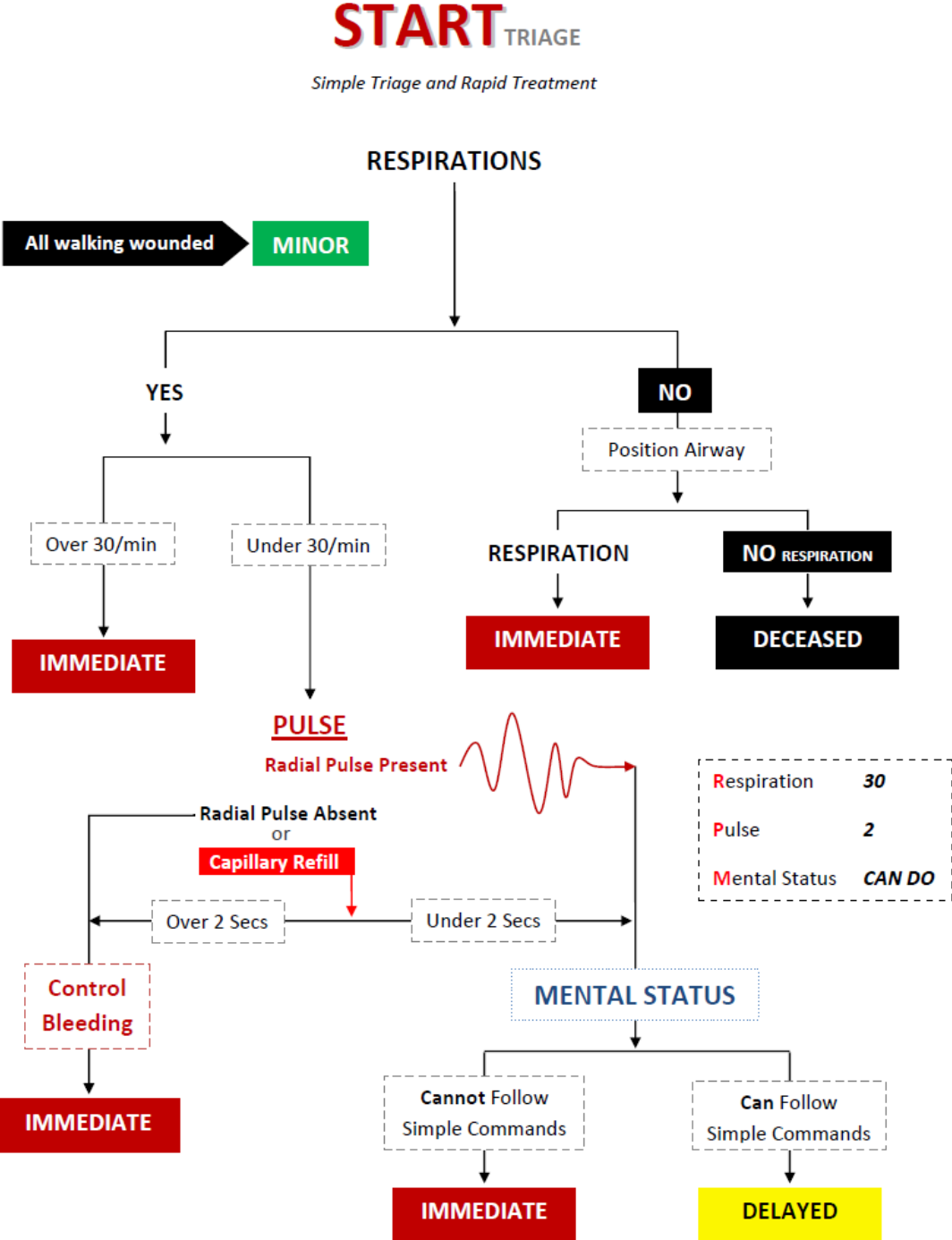


Figure 7.1: Algorithm for TRIAGE

System Analysis and Architecture

Work Flow Model

The tool allows for two roles in the system, namely

- 1) Administrator (or admin) and
- 2) User/Team

In the tool, 'Cases' are created by the admin by entering medically relevant description and details on respiration, position airway, pulse, mental status, correct tag and if possible, a related picture. 'Questionnaires' are then created from a subset of all cases present in the database. An 'Exercise' is created by selecting a questionnaire and assigning it to a subset of users/teams registered and willing to participate in the exercise. The admin workflow and the user work flow are presented below:

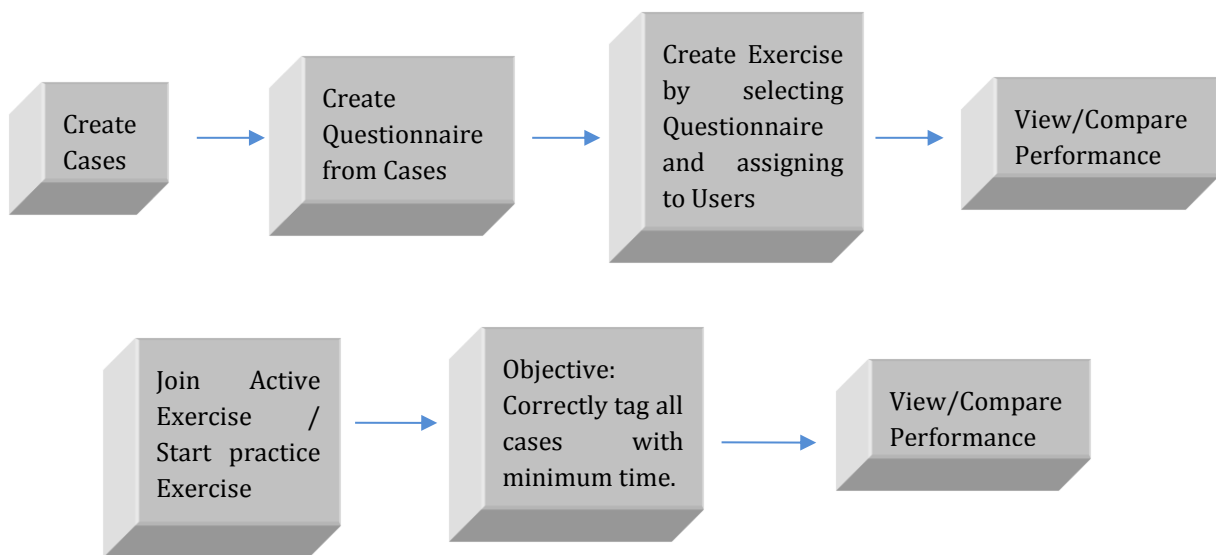


Figure 7.2: Work Flow Model for Administrators and Users

Both the admin and the users can view live results of an active exercise, including the individual and team performances in a classroom mode. The tool also provides 'history' of functionality for the users where they can view how they have performed in previous exercises. The user/team will also be able to practise a random exercise if an active exercise doesn't exist. The practice exercise is created randomly from the subset of cases in the database.

Use Case Diagram

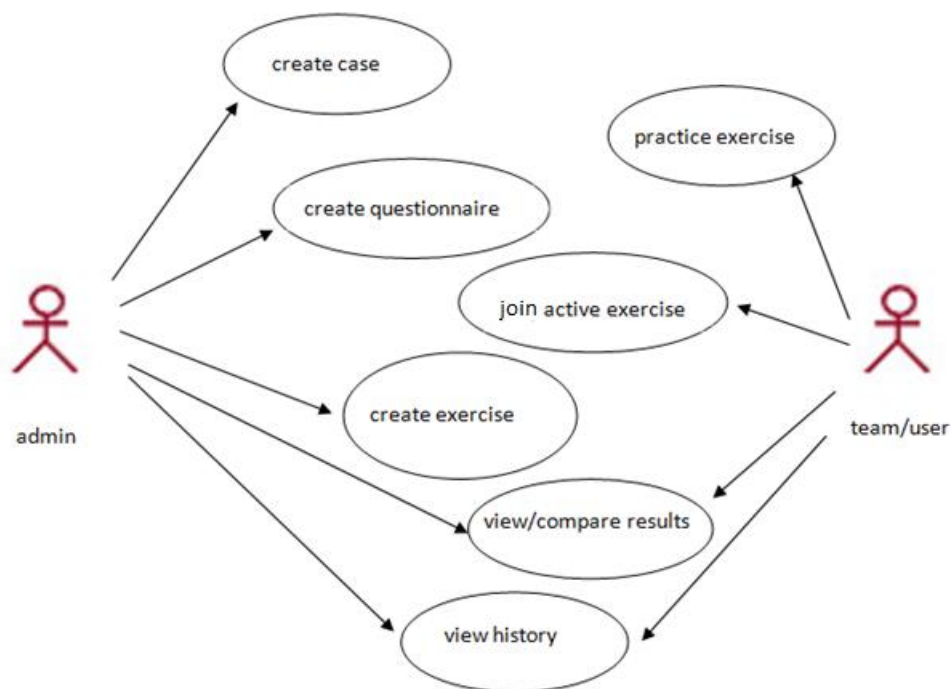


Figure 7.3: Use Case Diagram for the Triage Exercise tool

1. User (i.e Participant) use cases

i. Join Active Exercise

The user/team can participate in an exercise once assigned by the admin.

ii. Practice Exercise

The user/team will be able to practise a random exercise if an active exercise doesn't exist. The practice exercise is created randomly from the subset of cases in the database.

2. Admin use cases

i. Create Case

The admin can create a case by entering details such as description, respiration, position airway, perfusion, mental status, correct tag and relevant picture depicting the case.

ii. Create Questionnaire

The admin can create a questionnaire from a subset of existing cases in the database.

iii. Create Exercise

An exercise can be created by selecting a questionnaire and assigning to users/teams registered to participate in the exercise

3. Admin/User use cases

i. View results

Both admin and users can view the live results of an active exercise. They can see the performance of all teams participating in the exercise in classroom mode.

ii. View history

This is a common functionality for both the admin and the users where they can view how they have performed in previous exercises.

Entity Relationship (ER) model

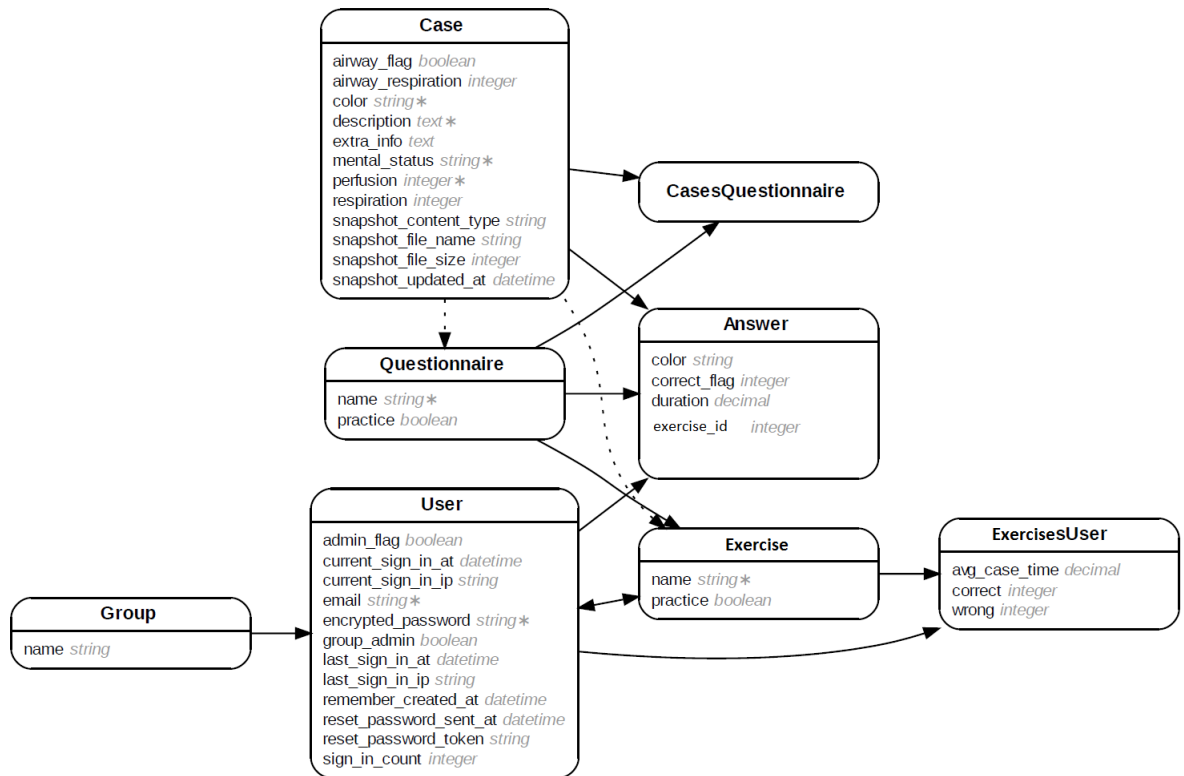


Figure 7.4: Triage Domain Model-ER graph

Beta Testing

The initial demonstration of the software prototype (Beta) was carried out at M S Ramaiah Memorial Hospital, Bangalore. Personnel from police, fire service, and medical departments participated in the exercise. They were divided into teams and were presented with a set of

cases to be tagged. The software, a desktop based tool, was developed on the *eAdventure* platform. The results of the feedback from different teams were assimilated and analysed. A web-based tool (possibly usable on portable, hand held devices) was considered necessary. So, a new Triage Exercise tool was developed using Ruby on Rails (ROR) as the web application framework. The web based classroom version of the tool was demonstrated at the Institute of Nuclear Medicine and Allied Sciences (INMAS), Delhi. The current version of the tool is fully automated and database-backed and provides live results of exercises using a polling model.

System Architecture

The basic Model-View-Controller (MVC) architecture used in the development of the tool is shown in Figure 7.5:

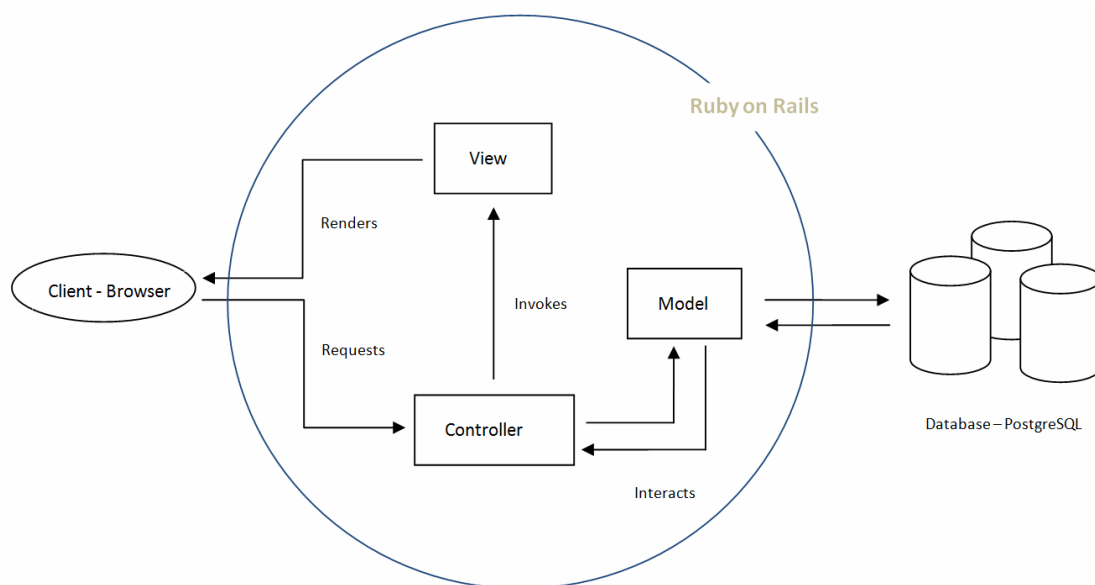


Figure 7.5: General System Architecture (Triage Exercise)

MVC is an object oriented design pattern which tries to divide an application into three components, namely the Model, the View and the Controller [2].

- **Model:** A model contains the applications business logic and represents the information (data) of the application and the rules to manipulate that data.
- **View:** The user interface is represented by the view. In a web based application the view is implemented as a template which renders an HTML page.
- **Controller:** The communication between the model and the view occurs via the controller. The incoming requests by the web browser are associated with controller actions which interact with the model for data and pass them on to the view for presentation.

The detailed system architecture is shown in the appendix to this chapter.

Ruby on Rails

Ruby on Rails is a web application framework for developing 'database driven' web applications. It uses "Convention over Configuration" concept which is well suited to 'agile' development. Ruby on Rails [1, 2] was chosen over other frameworks because unlike other frameworks, it

1. is open source
2. is suited for new code bases built from scratch
3. has a quick turnaround time
4. is preferred for "Mashup" (custom-featured) applications
5. works well for iterative projects, especially when developing early prototypes

jQuery

jQuery was used to develop the front end of the tool. It is a fast and lightweight JavaScript library which aids in rapid web development. The main advantages [3] of using jQuery are:

- Easy to use
- Large code library powered by strong open source community
- Ajax support
- Comprehensive documentation

The tool was developed and tested using SQLite3 as the backend while postgresSQL was used in production stage. The project went through multiple iterations with Extreme Programming (XP) methodology.

Further Enhancements

The tool can be modified to display videos. The same architecture can be used to develop a Multiple Choice Questionnaire or a timed quiz by modifying the fields.

Bibliography

[1] <http://www.exist.com/why-rubyonrails>

[2] http://guides.rubyonrails.org/getting_started.html

[3] <http://www.jsripters.com/jquery-disadvantages-and-advantages>

Appendix to Triage Exercise

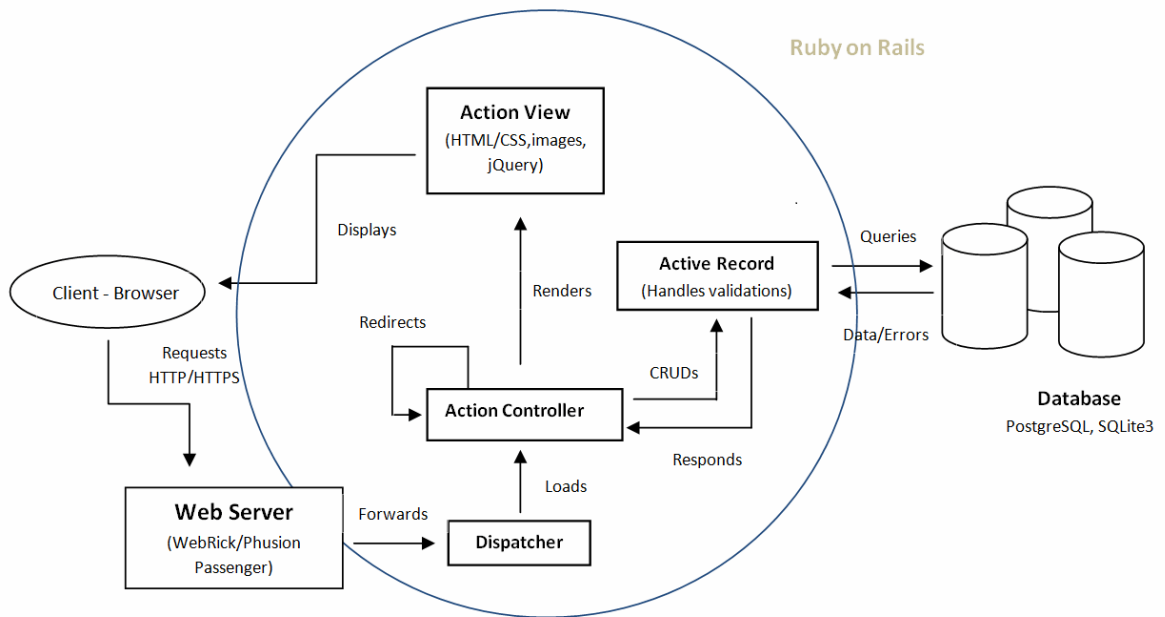


Figure 7.6: Detailed System Architecture (Triage Exercise)

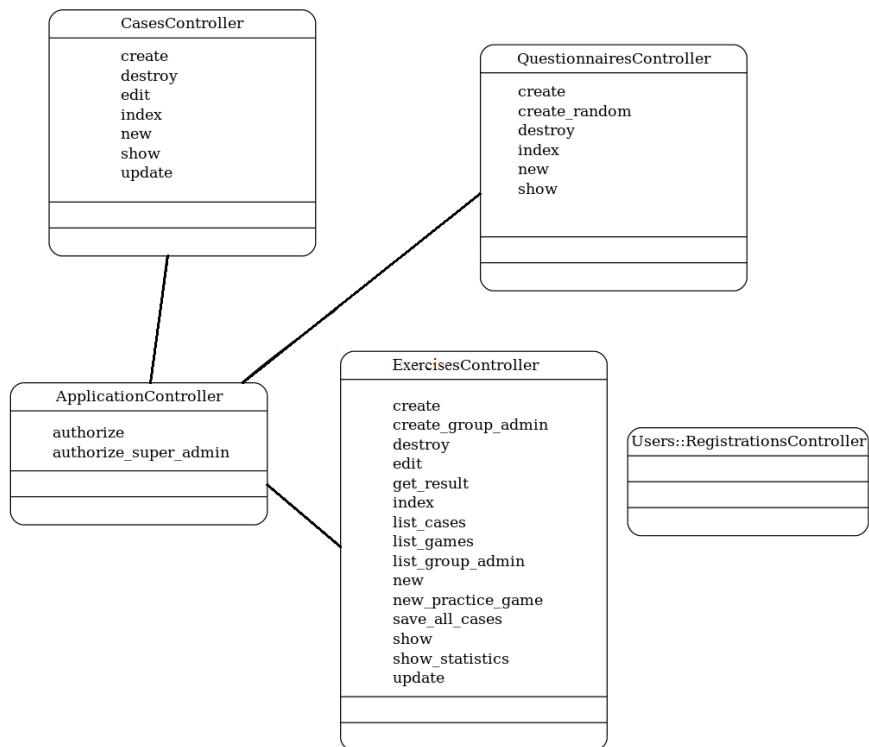


Figure 7.7: Class Diagram - Controllers

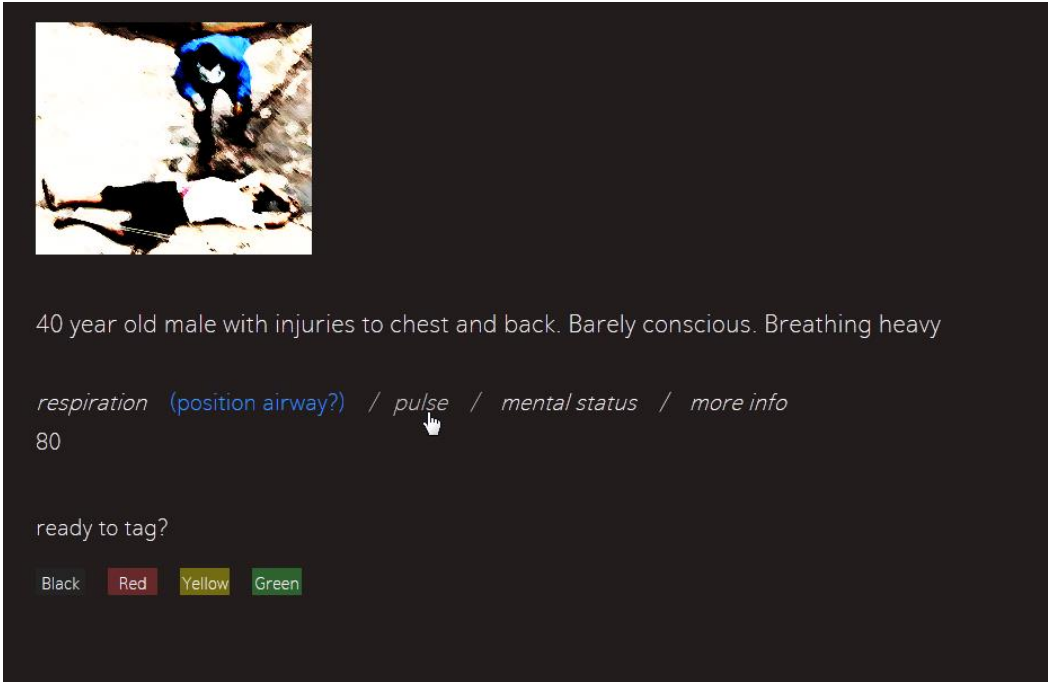


Figure 7.8: Triage Exercise in Action (User view during an exercise)

8. Multi-Agent Models

Multi-Agent Technology

Planning and training for disaster management will vastly improve the efficacy of decisions taken during a disaster and is likely to lead to a reduction in loss of life and property damage. Information technology, modelling and simulation along with gaming have been used to plan and train for emergencies. Disaster management is a complex problem, exhibiting *emergent properties* which are not easily tractable.

One way of studying a system is to break it down to its component sub-systems. These component sub-systems are then modelled as individual systems or 'agents'. The agents are allowed to interact with one another. The interactions result in some forms of emergent behaviour which are a subset of the actual emergent behaviour of the original system. The key advantage here is that individual subsystems can be observed in greater detail. The interactions and emergent behaviours expressed by the model are therefore tractable. In addition, the number of agents and their abilities determine the effectiveness of the simulation.

The current agent based platforms focus on the abilities of the agents, thereby providing a highly flexible agent programming interface. While this feature allows these platforms to contain a large variety of agents, the number of agents that can be created within these platforms is highly limited. This is due to the large overheads required by either the communication or the visualisation layers (or both). We have created a new platform which allows diversity within agents along with a large number of agents as well. It has a loosely coupled architecture and uses Java as the principle programming language. The platform is used to simulate a disaster situation in Bangalore.

Essential Parts of an Agent-Based Simulation Framework

An agent based simulation framework allows one to define agents i.e. to define the functions of the individual subsystems in a complex system. The functions involve the capabilities and the actions that these agents can perform. The framework is also responsible for the communication between these agents. Figure 8.1 shows the basic framework.

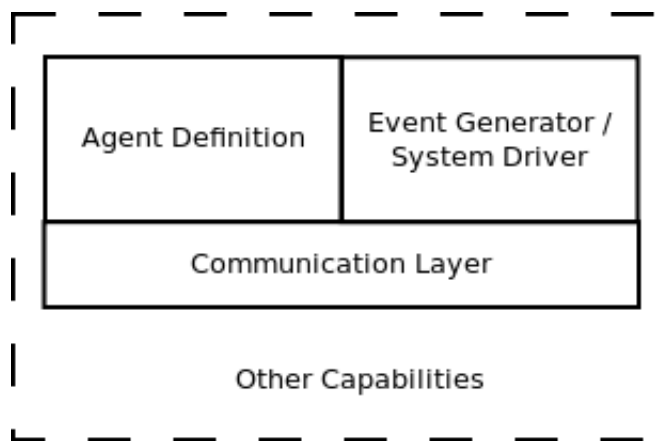


Figure 8.1: Conceptual Framework for an Agent-Based Model

Agent Definition Mechanism

Most agent-based simulation frameworks provide a template for defining the type of agents. An agent of a particular type has a set of goals and can perform defined actions. Also, at any given time, there may be different types of agents and a number of individual agents of the same type present in a system.

Agents can perform a set of defined actions which changes the state of the system. The agents perform these actions to achieve their goals or final states. The choice of action depends on the type of agent. Thus, by defining goals and varying the actions, different types of agents can be defined. The agent platform is responsible to create the specified type and number of agents.

Event Generator or System Driver

The event generator or system driver module generates signals at appropriate intervals to drive the entire system forward. The signals can either be clock signals generated at regular intervals, or they can be generated after the system is in intermediate states. All agents are aware of the signal once it is generated and react to it depending on their description.

Communication Layer

Messages represent the interactions among agents and thus are responsible to bring about group behaviours. Emergent behaviour may also be observed on account of communication agents. The communication layer is used to relay the event signals to all the agents. The agents are free to choose any protocol they require for communication. The function of the communication layer is only to transfer a message from one agent to one or more agents without interpreting the message. This is a key requirement as this allows maximum flexibility in communication.

Visualisation or Output Module

The visualisation or output module displays the result of the simulation using visual aids such as graphs and images. The visualisation shows us the various interactions taking place between the agents. It will also put into perspective how the interactions and behaviours of agents translate to an emergent behaviour.

Some of the other agent framework capabilities include:

1. An ontology framework
2. Yellow page service to locate agents
3. Agent transportation capability to move agents among frameworks
4. Logging and tracing agents

These capabilities although useful (and sometimes required) are either implemented using the essential modules mentioned above or developed as an extension. By keeping the essential modules flexible the framework can be extended to include other services. The next section elaborates on the requirements and constraints placed on the agent frameworks for large scale simulations.

Architecture of the Agent Platform

Overview

The framework has a distributed architecture. It provides an infrastructure to create agents with specific behaviour, messaging, Keyhole Markup Language (KML) generation and data storage. A Care Taker Agent (CTA) is the main entity that runs on a machine (node).

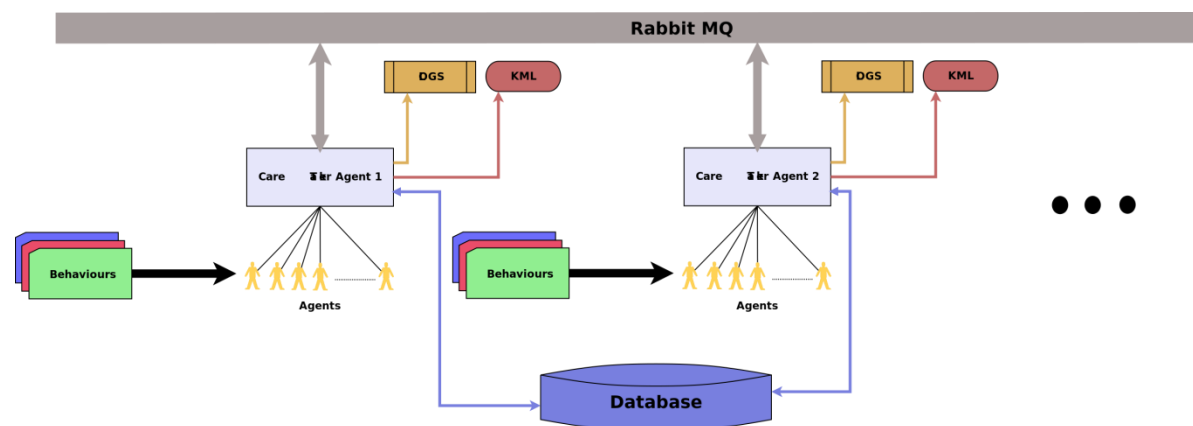


Figure 8.2: Detailed Framework for an Agent-Based Model

The components of the framework are

1. CTA (Care Taker Agent)
2. Agents
3. Message Queues
4. Datastore
5. KML utility

Care Taker Agent

Care Taker Agent (CTA) is a container within which agents live and execute (or perform) their behaviours. All agents within a CTA are necessarily of the same type. CTAs communicate using asynchronous message queues. A CTA is responsible for the following:

1. Agent creation
2. Inter agent messaging
3. Inter CTA messaging
4. Writing simulation output
5. Shutting down the simulation locally

Agents

An agent represents a real world or abstract entity who is a vital actor in the simulation. An agent has a list of attributes which define what the agent is and how it is different from other similar agents.

Agent Attributes

Agent attributes are abstract classes. For a particular type of agent (e.g. Person) the class has to extended and required set of attributes defined. The values of the attributes could be stored in a database or a XML file.

Agent Behaviour

Every agent has at least one objective. To achieve its objective a agent has to undertake certain actions to manipulate the data, interact with other agents etc. This is done through actions called behaviours. An agent may have one or more behaviours which are executed in a defined order.

Messaging

For a distributed system, the messaging subsystem (or communication channel) is a vital component which facilitates information sharing and synchronisation. The framework the communication is entirely confined to CTAs. Individual agents can communicate using its CTA as proxy. The messaging system should satisfy the following requirements:

1. High throughput - A few thousand messages should be delivered, end-to-end, within a second.
2. High availability - The system should not crash under high load.
3. Asynchronous messaging - Synchronous messaging will slow down a large system. An asynchronous technique which works like mail boxes is desired.
4. Platform independence - It should not be tied to a particular programming language or an operating system. This enables us to build heterogeneous CTAs.

The types of messages we use are:

1. Data e.g. agent attributes, position of emergency vehicles, traffic signal state etc.
2. Timing messages e.g. I am finished with the current tick.
3. Shutdown e.g. I am finished with work.

[RabbitMQ is our message queuing system of choice; it is based on the AMQP standard and is an open source under Mozilla Public License. RabbitMQ has a client's interfaces for many platforms - Java, .Net, Python and others. It supports multiple schemes of communication including 1-to-1, 1-to-many, store and forward, file-streaming etc.]

Datastore

The simulation requires a datastore to store:

1. Agent attributes and the data input
2. The output of the simulation

The framework interfaces with a GIS \index{GIS} database (Postgresql with PostGIS. Other databases could be used by changing the appropriate properties files and drives.

Output Module

The output should be visualised so that it can be analysed. In our framework the output visualisation module is decoupled from the simulation engine. This is done to achieve the following

1. Enable visualisation on multiple remote clients
2. Enable visualisation on thin clients so that there is less processing overhead on the remote machine
3. Replay the same output as and when required

The output of our framework is in the form of a KML file which can be visualised on Google Earth. Google Earth is 3D virtual earth software which supports visualisation and animation of spatial data using KML files. The KML format contains spatial data and presentation information.

Implementation Details

The framework is implemented in the Java programming language. Java provides easy of programming and has a wide range of helpful libraries. Configuration details such as number of agents, the geographic bounding box etc are stored in text files.

Care Taker Agent

A Care Taker Agent is an abstract class which implements Queue User. The main logic of Care Taker Agent is as follows:

Algorithm 1: The Care Taker Agent Operation

- 1: Initialise
- 2: Create Agents
- 3: while there exist agents with unsatisfied objective do
- 4: Wait for all other CTAs to signal 'done'
- 5: Run the agents
- 6: Write output
- 7: Message other CTAs 'Finished with current tick'

- 8: end while
- 9: Message other CTAs 'Finished with work'
- 10: Exit

QueueUser enables CareTakerAgent to receive messages from its message queue.

Agent

Agents are implemented as Java threads. An agent is an abstract class with the following attributes:

1. AID - unique identifier for the agent.
2. objectiveFlag - whether objective satisfied or not
3. statusFlag - true = finished doing work for this tick
4. statusFlag - false = did not finish doing work for this tick
5. composite behaviour - one or more behaviours

Being a 'Thread', an agent has the 'run' method where most of its logic is located. Calls to behaviours are made here.

Behaviours are implemented using the composite design pattern. Behaviours is an abstract class with an abstract method called 'run', [method signature: *public void run (AgentAttributes agentAttributes)*]. The parameter agentAttributes defines the current state of the agent. A person agent will have PersonAttributes like health, speed, best path etc. AgentAttributes is an abstract class; each agent type will have its own implementation e.g. PersonAttributes, VehicleAttributes etc. Every agent type will have its specific behaviour implementation, e.g. PersonMoveBehaviour, VehicleMoveBehaviour etc.

Messaging

Each CTA runs in its own JVM and inter CTA communication is through RabbitMQ message queue. Every CTA has an input queue - identified by the hosts IP address and a queue name. To communicate, a CTA has to write the message to the recipient CTAs queue. A CTA is notified as a new message is received – the *receivedMessage (Message message)* method is called.

Classes

Message.java

A message is a serialisable class with the following fields

Main attributes

1. type - int, helps to identify the type of message
2. content - Object, can be any serialisable Java object
3. sender - string, host name of the sender

QueueManager.java

Manages sending a message to the recipient and listening to incoming messages

Main attributes

1. queueUser - the CTA who is using this
2. queueParameters - the parameters of the input queue for the CTA

QueueUser.java

A class which wants to receive messages on an input queue should implement this interface. Observer pattern is used here. The CTA registers with a QueueManager and on receiving a message the queueManager calls the receivedMessage (message) method of the CTA.

Main methods

```
public void receivedMessage (Message message);
```

QueueParameters.java

A class to hold the parameters of the input queue.

Main attributes

1. queueName - the name of the queue
2. username - the username to access RabbitMQ
3. password - the password

Datastore

Since the Framework supports modelling/simulation in spatial domain, we required a Datastore which could support geometry data types and spatial functions - union, intersect etc. Postgresql is an open source object relational database, and the PostGIS plug-in empowers it with spatial features. The pgRouting plug-in adds routing and shortest path algorithms. The c3p0 library is used for connection pooling, so that the database can support a number of parallel connections from a particular CTA. In case one needs to change the data store the following changes are to be made

1. Edit properties in db.properties file located in database.connection package.
2. Add the required driver to the \\lib directory and add the same to the CLASSPATH
3. Create a Java class in the database package and write the queries in it.

Classes

Database related code is present in the src/database package. The src/database/connections package has the files for connection pooling and database configuration.

Sql.java

This is a static class with has implementation of all the required queries.

KML Utility

The utility is implemented using JAK (Java API for KML). It takes an ArrayList as points and draws them as place-marks. It can be used to create animation with points at different location at different timestamps. KML is a XML format and JAK is a parser which facilitates creating and editing the XML. JAK allows export of the XML into a text file, which can later be used for visualisation.

Classes

KmlUtility.java

Main methods

1. *public void addPlacemarks(ArrayList<Location> latLons)*
2. *public boolean writeFile(String filename)*

Geographic Information Systems

The framework is capable of simulation agents in a spatial context. This helps to program the following types of simulation to list a few

1. City traffic
2. Crowd simulation
3. Ecology
4. Material movement

The main requirements to incorporate spatial data in the framework are

1. GIS enabled datastore
2. Geometric datatype support in the programming language

GIS Datastore

The GIS stack is as follows

1. Database - PostgreSQL
2. Spatial support - PostGIS plugin
3. Routing support - pgRouting plugin

Geometric Datatype Support

Geometric datatype support is through wkb4j and postGIS Java libraries. WKB (Well Known Binary) is a binary format to represent geometric data; WKB4J is designed to read the WKB from a data source (e.g. PostGIS) and transform this data into corresponding Java objects. The postGIS library allows to store and manipulate geometric objects in Java.

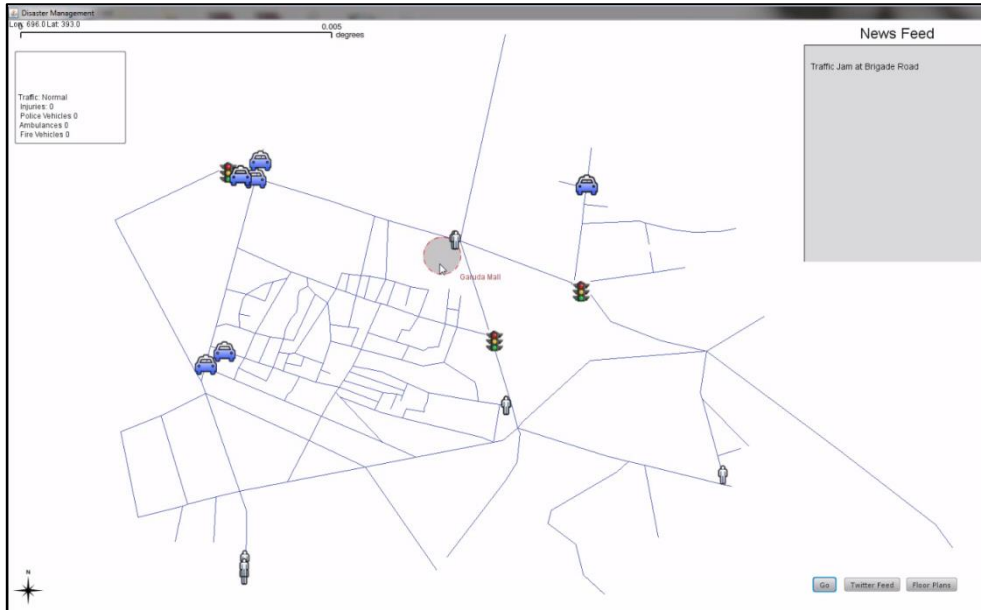


Figure 8.3: Multi-agent Model Initial View

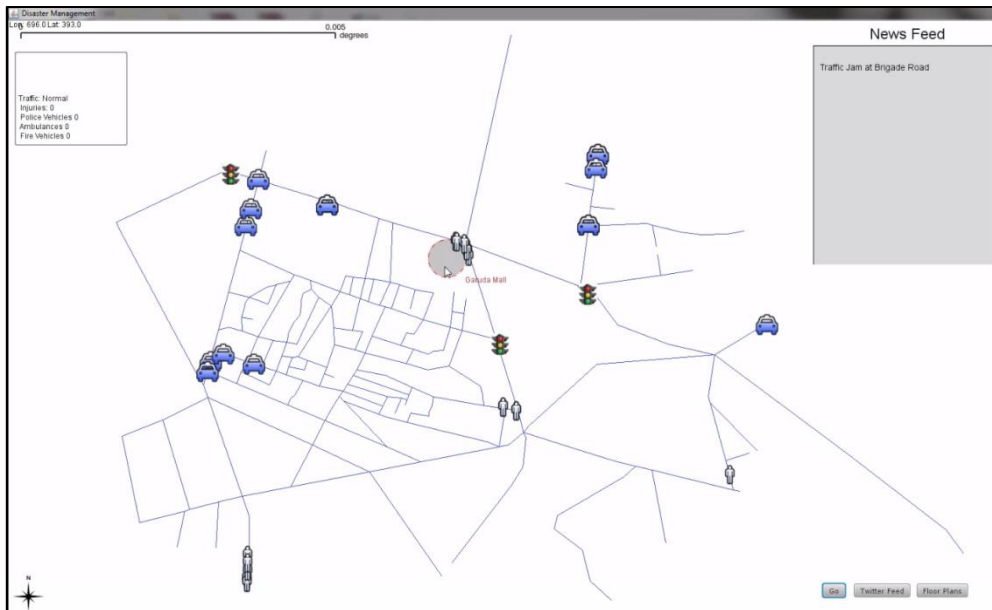


Figure 8.4: Multi-agent Model Traffic Increase

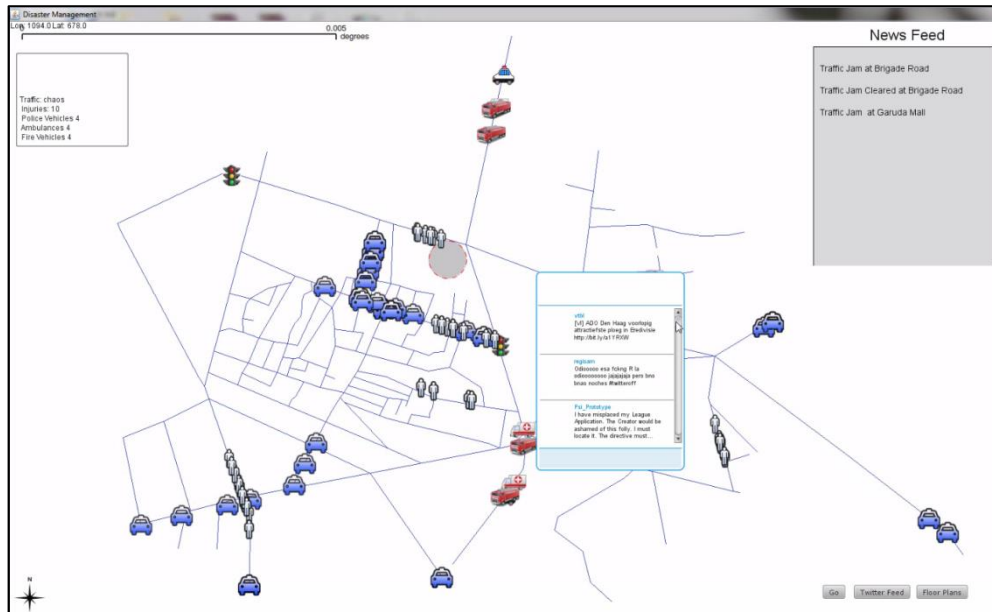


Figure 8.5: Multi-agent Model Emergency Traffic Re-routing

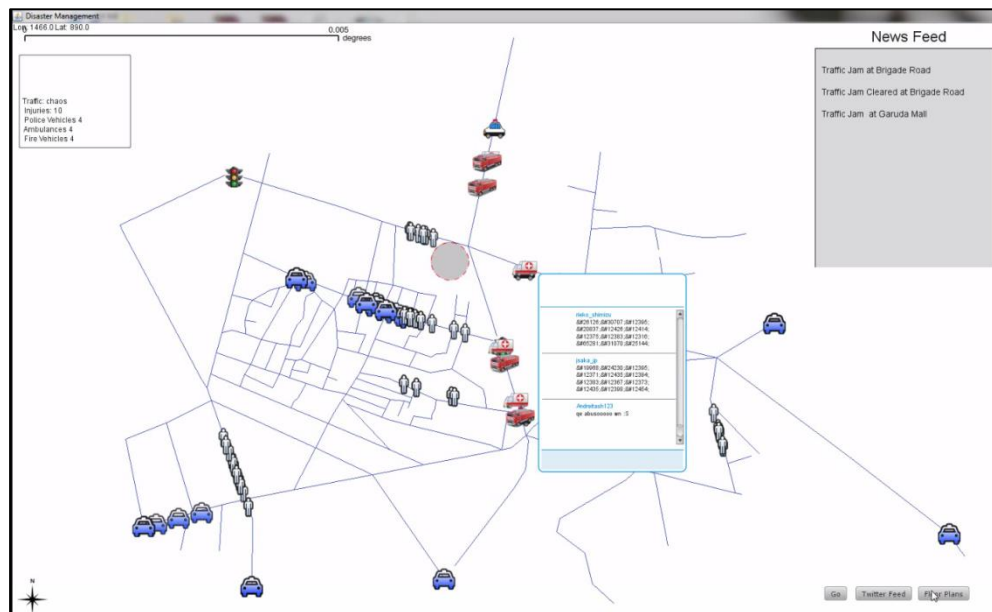


Figure 8.6: Multi-agent Model Emergency Vehicles Converging

Open Source Components

The agent-based platform was developed during the Phase-I using open source and free technologies like Java, PostgreSQL, PostGIS, RabbitMQ, Ubuntu, Google Earth etc. A disaster scenario was implemented on the platform to understand the movement of people, vehicles and emergency services. The output is viewed as animation in Google Earth.

9. Carlton Towers (A Case Study on Emergency Management)

"Carlton Towers is burning and six of us are trapped inside. The fire's above but there's smoke everywhere. Saw people jump to their death."

"Our rescue ladder. It'll only reach the fourth floor. We're on the fifth."

"Massive crowd outside. This must have choked traffic for kilometers around."

Thus began a series of tweets by techie Kiran Jonnalagadda (Twitter handle: jackerhack) [1], describing the start of what is now known as one of the worst disasters in Bangalore. The unprecedented fire at Carlton Towers, which caused the death of nine people and injured sixty eight, is filled with narratives of flouted safety norms, poor planning, shoddy infrastructure and a group of Emergency Management Systems (EMS) authorities overwhelmed by the sheer magnitude of the problems created in its wake.

About Bangalore

Bangalore is a teeming, thriving metropolis; with a population of almost ten million in the southern state of Karnataka. The heart of India's burgeoning IT industry, the erstwhile "garden city" has transformed into an urban landscape, with a swanky airport, a new Metro, new buildings and loads of construction everywhere. The population grew by a startling 48% during 2001-2011. It has 4.4 million vehicles with approximately 25,000 being added every day. Sadly, despite efforts, the infrastructure growth has been totally inadequate. This unplanned growth has created cramped spaces, roads with potholes, and massive traffic jams, making it difficult to handle disasters.



Figure 9.1: Traffic Jam on Mahatma Gandhi Road in Bangalore

About Carlton Towers

Carlton Towers, a seven-floor concrete and glass office structure, was built in 1999 on Old Airport Road, Bangalore, on land owned by Anurag Jain. (Jain also owned the office units on the second, third and fourth floors of the building and was a co-developer). It has two buildings.



Figure 9.2: Carlton Towers, Bangalore

Soon after it was built, the property was sold to 94 companies and individuals as office spaces. The tenants included famous Indian and multinational companies like Mid-Day, BOSE Corporation, Indi Joe, Star Worldwide Group, Celestial Systems, IAL India Ltd., Hanmer & Partners, Indo-Nippon Foods Ltd., Discovery Networks India and Hamilton Realtors [4]. The ground floor had popular eateries like The Bombay Post and TGIF, along with a Reliance retail hyper mart. It was estimated that around 1500 people would be present inside the building on any working day.

Carlton Towers was issued a No Objection Certificate (NOC) under the National Building Code 2005 in 1999 by the fire department [3]. Subsequently, the Bruhat Bangalore Mahanagara Palike (BBMP) issued an Occupancy certificate.

The Fire (and the Initial Response)



Figure 9.3: Smoke Coming Out of Carlton Towers in Bangalore, 2010

The fire broke out on the first floor at 3:00 pm on Tuesday, February 23, 2010. It started from the lift shaft on the second floor. A software firm in the building, owned by S K Suresh Kumar (with offices in unit numbers 111 and 113), during the course of renovations to its offices, had replaced high-quality copper cables with low-quality aluminium cables. The cables had lost their capacity to carry electric current and caused a short circuit, which in turn caused the insulation to burn. First the smoke was localised in the centralised cable duct and then moved to the lift shafts [5]. More than the fire, it was the dense, choking smoke that led to a sense of panic. Those on the lower floors managed to walk out of the building or were helped out. One hundred and fifty people needed evacuation from the upper floors. Nine people, three of them women, died of either asphyxiation or jumping from the floors. Three people, encouraged by onlookers holding purported “safety nets”, leapt to death in panic by breaking open windows on the sixth and seventh floors. Overall, 68 people were treated for injuries, with some requiring hospitalisation. The fire was confined to only one of the buildings, but both were evacuated. Twenty two fire engines were summoned to Carlton Towers. The first few reached within 15 minutes of the first emergency reported but many were hampered by traffic congestion and could not get there till much later [7].

Violations, Faults and Failures

There were both major and other failures/violations which hampered the rescue operation. The major ones included:

1. All exit doors were locked or barricaded with grills on all the floors [6]
2. Smaller exits and several owners or tenants had blocked the access to their individual areas.

3. Unused furniture and material were stacked in common areas like stairwells and corridors.
4. Safety nets used by the fire-fighters collapsed when the victims jumped from the floors (based on an eye witness account). However this has been refuted by the fire-fighters [7]. Three people, who jumped, died due to injuries sustained.



Figure 9.4: Panic during the fire at Carlton Towers in Bangalore, 2010



Figure 9.5: Rescue Attempts during the Fire at Carlton Towers in Bangalore, 2010

5. Insufficient clearance between the compound wall and the building blocked the entry of the fire engines (possibly carrying longer ladders). The space was expected to be six metres wide but in reality was reduced to two and a half metres [10].
6. The water tanks for the internal fire-fighting system (sprinklers) in the building were dry.

Some other violations and failures included:

1. Ten barrels of diesel were stored on the roof for use in generators. Although these barrels had no direct effect on the fire in this incident, the consensus amongst experts was that if those barrels had caught fire, then there would have been a far greater catastrophe
2. No proper breathing apparatus for fire-fighters, which led them to take five minutes breaks to clear their lungs
3. Massive traffic jams due to crowds and onlookers and the evening rush-hour that didn't allow the larger "turn-table" based fire engine and ladder to reach the site
4. The first few fire tenders being low on pressure and the lack of water hydrants and
5. Lack of basic fire safety education amongst the general populace.

Observations and Recommendations

In light of the failures, the following policy changes are advocated:

1. Conducting appropriate fire drills should be made mandatory for all urban structures and include emergency evacuation procedures (currently mandated to be three minutes to exit the building)
2. Train residents/tenants to use fire extinguishers and read and understand exit route maps, and improve communication exercises among the responders. CSTEP has developed a Coordination Protocol Exercise to facilitate this kind of training.
3. Maintain an inventory and map all the emergency management assets (such as sprinklers, fire extinguishers, first-aid kits etc.) and make it available to all the stakeholders. A testing schedule must be part of the documentation.
4. The first fire-engines reached the spot in fifteen minutes. However, as described earlier, the engine with the turn-table based ladder was unable to reach the site due to traffic congestion. CSTEP researchers have created an allocation and routing prototype that not only decides optimal allocation of EMS resources but also calculates the shortest path (and therefore response time) to an emergency. This prototype will also factor in traffic patterns (flow, density) in calculating the minimum amount of time needed for a resource to get to the location of incident.
5. Fire-fighters complained about lack of access within the building. One possible reason for confusion could be poor building blueprints and/or diagrams. With these, it would have been possible to quickly decide on alternative routes. CSTEP researchers have developed a system to convert 2D plans into 3D models and walkthroughs that give first responders a better idea of layouts and exit routes in complex urban structures.
6. Plan and conduct Fire Safety and Emergency related education for the general populace. One way might be to institute a mandatory policy (for example as part of the

new employee induction program) to have a training programme on fire safety at the work location.

Epilogue

Following the outrage at the handling of the incident by all agencies involved, a series of measures were taken by courts, government administration and the public at large. Some of these were:

- Following the fire, the High Court of Karnataka had directed the Department of Fire and Emergency Services to immediately take up fire audit of the city's high rises and complete it by December 2013. As of February 24, 2013, about thirty percent of structures have not been audited and are without NOCs. Also, there are only 7 or 8 water filling points in and around Bangalore [9].
- Two skylifts (truck mounted platforms) that can reach a height of 54 metres have been procured from Finland at the cost of Rs. 10 crore to help the fire department deal better with about 800 fire incidents are reported in Bangalore every year [8].
- Six people have been arrested in this case. They are currently out on bail and the cases have been booked.
- 'Beyond Carlton', a support group initially formed by the survivors and family members of the people killed in the fire, has expanded membership to encompass numerous citizens who are concerned about fire safety [11].
- Finally, after numerous repairs, Carlton Towers received an NOC from the fire department by March 20, 2013 [12]. However, the Electrical Inspectorate is yet to give its assent on the reopening of the building.

Carlton Towers still stands, its walls a grim reminder of the lack of disaster management expertise in Bangalore.

References

1. <https://twitter.com/jackerhack/status/9521471028>
2. <http://bescom.org/wp-content/uploads/2013/01/Paper-ClippingEnglish-15-03-2013-1.pdf>
3. <http://www.hindustantimes.com/India-news/Bangalore/Carlton-fire-could-ve-been-bigger-tragedy/Article1-512593.aspx>
4. http://www.ijnm.org/media_uploads/thesoftcopy/2011_2012/city_carlton.html
5. <http://www.bangaloremirror.com/index.aspx?page=article§id=1&contentid=2013031520130315010048935a2e6724#>
6. <http://www.dnaindia.com/bangalore/1352748/report-carlton-towers-was-anything-but-rescue-friendly>
7. <http://www.ndtv.com/article/india/bangalore-fire-sprinklers-smoke-alarm-didn-t-work-16809>
8. <http://www.hindu.com/2011/04/14/stories/2011041461030200.htm>
9. <http://www.thehindu.com/news/cities/bangalore/a-third-of-citys-high-rises-have-no-nocs-from-fire-department/article4446961.ece>
10. <http://www.hindustantimes.com/India-news/Bangalore/Carlton-Towers-flouted-safety-rules/Article1-512720.aspx>
11. <http://beyondcarlton.org/>
12. <http://www.bangaloremirror.com/index.aspx?page=article§id=1&contentid=2013020120130201090420298c01abc08>

10. Plan for Future Work

Emergency and Disaster Management is essential to ensuring the safety of the citizens and in protecting the assets of a nation. During the course of the projects (SIMPLANEMS and SIMPLANEMS-II) it became evident that issues pertaining to technology and policy for disaster management are intertwined and need to be examined further.

The broad topics that warrant further examination are:

1. Agent based modelling and simulation:

Emergency and disaster management is a complex system which needs to be examined with appropriate tools. Agent based modelling is one such tool which examines the systemic emergent properties of many interacting agents. The ability to simulate crowds and assess suitable interventions for responding to emergency situations is an important and possible area of future research.

2. Use of computation and visualization for the development of a decision-analysis and research platform:

With the advent of computer technologies, analytical planning models have become the main stay of complex decision-making. Improvements in computation speeds, coupled with emergent artificial intelligence and data-mining technologies, have dramatically changed the process of decision-making for complex high-level defence and civilian planning. With better networking technologies in place collective decision-making, supported by huge datasets, is imminently possible. Another area of research that can be pursued is the development of a decision-analysis and research platform that uses supercomputing facilities, and a plethora of planning models in order to explore various trajectories of decision-making.

3. Development of tools for training and testing of personnel in emergency and disaster management protocols and procedures:

Training of emergency response personnel (be they civil security staff or common citizens) is essential for effective response in times of emergencies and disasters. Traditional training mechanisms such as class-room instructions, workshops and mock-drills have limited effectiveness and concomitantly high costs. Through the development of the Triage and Communication Protocol Exercises, we have determined that certain types of education and training can be done in a distributed yet relatively low cost manner. The development of such training exercises to be conducted for a vast number of persons across different geographic locations is another area of future work that may be pursued.

The research ideas listed above are useful to any entity interested in security – be they the military, civil defence (NDMA), civil administration or agencies such as hospitals that are called upon to assist during an emergency. It is a rich and rewarding area for future research.



Center for Study of Science, Technology and Policy,

Dr. Raja Ramanna Complex,

Raj Bhavan Circle,

High Grounds,

Bangalore - 560 001

Tel: +91 (80) 4249-0000

Fax: +91 (80) 2237-2619

admin@cstep.in

www.cstep.in