Development Optimisation Engine

January 27, 2009

We are a group of friends at a research-oriented institute at Chennai, CMI, and are passionate about developing software to make a change in the world around us. This document is an attempt to describe our dreams and the challenges we hope to solve.

Amit Prakash Ambasta	amit.prakash.ambasta@gmail.com	M.Sc. C.Sc.
Abhishek Chhajer	abhishek.chhajer@gmail.com	M.Sc. C.Sc.
Arpith Siromoney	arpith.siro@gmail.com	B.Sc. (Hons.) Phy.
Hrishikesh R. Terdalkar	terdalkar18899@gmail.com	B.Sc. (Hons.) Math. & C.Sc.

The Challenge

Development in countries such as ours is characterised by a lack of involvement of the common man, at the grass roots level. While technology has reached the people, it is our belief that we are not yet empowered enough to influence the decisions that affect us on a day-to-day basis. It is these problems, we feel, that technology should help solve – for a brighter India tomorrow.

Accident-relief, for example, is hampered by a lack of awareness of an optimal strategy towards utilisation of existing resources. Mobilising ambulances and health care personnel who are nearest to scene can help improve survival chances. In the same light, quicker emergency notification leads to quicker response times.

Another, though seemingly different, example is the mainenance of critical resources in supply logistics. If a sewer fails to function, or a street light goes dead, or, in a more drastic situation, a terrorist attack wipes out commodity supply to a city, then quick notification to those whose job is to fix things can go a long way in terms of restoration of the supply. Instant notification, however utopian, is unlikely to lead to an improvement unless optimised strategies are used towards the deployment of reserve resources. This is where we feel technology, and our dreams, can come together.

Our Dream

As we spent long nights discussing the problems we could see around us, it became increasingly apparent that there was a common thread that ran through the seemingly different issues. Increased involvement at the grass-roots level would only be possible through pervasive technology that was already available to the people who were affected by the problem, and faster solutions could only be enabled if new ideas were used to help optimise development through mobilisation of resources.

What we are working towards, is a distributed system that uses SMS and web-based XML-requests to alert a centralised server. The server-client model allows for multiple user agents to be utilised in feedback for the engine. The engine can then perform its optimisations on the users' data via plugins that are managed by the ServiceManager. A WidgetManager allows multiple front-ends to use consistent interface paradigms, in the form of widgets.

How technology steps in

A plethora of tools aid us in our quest: notably, we use the .NET framework, SQL Server, Interoperability techniques (JNBridge Pro), XML-based transactions, SMS Server, Silverlight, and ofcourse, Microsoft Visual Studio as our trusty IDE.

> All problems in computer science can be solved by another level of indirection

We abstract the challenges that we wish to solve, from the techniques that are used to solve the foundational problem. In this way, our Development Optimisation Engine can be extended easily by developers who purchase the software to help solve the problems that challenge their customers, clients, and friends.

The first abstraction, therefore, is the fundamental objects that each challenge deals with. For instance, municipal planning authorities have to deal with sewers, garbage disposal units, public transportation, and lighting nodes. On the other hand, hospitals need to keep track of the ambulances they own, the services they can access when they have no solution on their own, and most importantly, the people who need their aid in times of trouble. Objects can be defined either in terms of building blocks (such as nodes and links) or as instances of classes that can be registered with the Object Repository [1].

Of course, buses have to travel from point A to point B, and this is where the ServiceManager [2] comes in. Developers can further extend our software by programming services that utilise the objects stored in the ObjectRepository. These services can then be run as daemons which, when registered, can be called by the ServiceManager when needed. Asynchronous design prevents the engine from having to wait on a service which could have, like our buses, stopped running. To retain common interface paradigms, services can use widgets that are registered in the WidgetBank [3]. Microsoft Silverlight lends advanced usability to both web and desktop interaction with representations of, for example, drag-able buses and explore-able maps.

(insert diagram here)

The last component of our architecture is the Engine [4], that grants services access to manipulate user data. This service oriented model enables the engine to utilise developer designed algorithms that are specific to the problem at hand, while retaining the usability of the plugins that are pre-designed, which solve the challenges we are most interested in.

As described before, SMS and web based interfaces [5] are used to provide alerts to modify nodes in the data that the engine is handling, dynamically. And needless to say, the engine conducts its transactions with the services via XML [0].

Let's take a look, for example, at how our software can be used by health-care providers in rural India. We have developed objects to represent health centers (like hospitals), mobile care providers (like ambulances) and widgets to represent them. The map object enables users to view the routes and positions of the nearest care provider. The web interface is complemented by the SMS service, which allows users to alert the Engine of their location and request assistance. We are developing widgets to represent the objects in Silverlight. Several algorithms have been implemented as services, and the Engine can

(insert the actual working of the engine)