A Meta Model for Multiple Transaction Models

Author: Justin Spiteri Supervisor: Patrick Abela

Project Background

It has been widely recognized that traditional transaction models with ACID (Atomicity, Consistency, Isolation, and Durability) properties are generally not applicable to transactions which have a compound nature. This has led researchers both in industry and academy to create of a series of specialized models which cater for this type of transactions, each oriented towards a particular range of applications, according to the needs of the particular developer.

The main problem with these models is that unlike traditional transaction models, where one model suited a very wide range of applications, these models fit only a narrow band of applications to the extent that in some cases, a completely customized model must be developed for an application. This is due to the fact that while ACID based models are based on a fixed workflow consisting of two parties, these models must cater for multiple parties, possibly resulting in a different workflow for each application, thus requiring a redesign of the model every time.

At present, to redesign a model, a developer must have a solid knowledge in the field of transaction management, and apprehending all the concepts needed is a time consuming task. While standardization efforts have been made with workflow modeling languages such as BPEL, these still require the developer to learn the transaction oriented language syntax in order to be able to create adequate transaction models.

Project Objectives

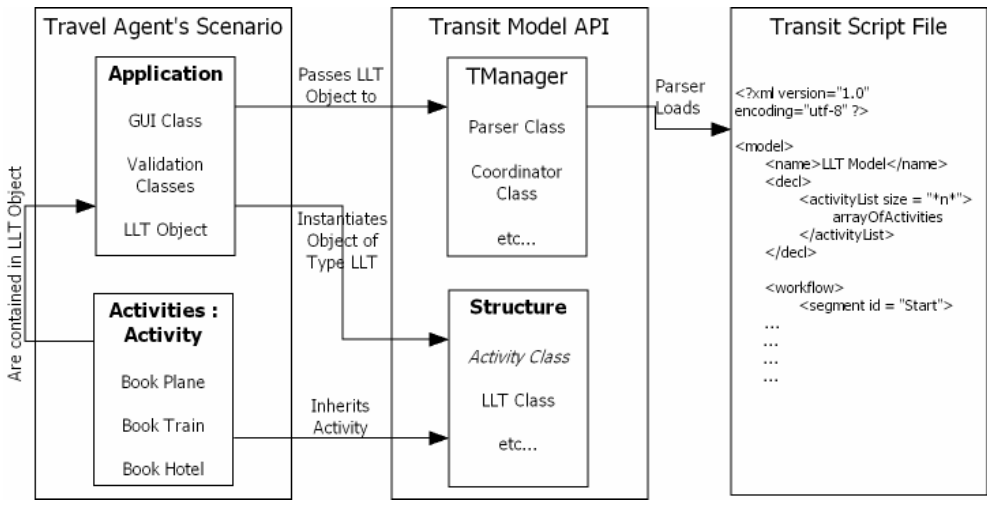
The main objective of this project is the introduction of a series of novel concepts resulting from an extensive research which standardize the process of transaction model definition, through the design and development of a specialized Meta Model, the Transit Meta Model.

The Transit Meta Model is an XML based scripting language which allows the definition of any workflow based transaction models using classic imperative language blocks such as “if then” and “for do” statements. Being compliant to W3C XML specifications, and having well known language constructs, the language succeeds in simplifying the transaction model description process, by virtually abstracting the developer from complex transactional details such as transaction inter dependencies. The only knowledge needed by a developer to use the Transit Model Solution is basic OOP programming language knowledge, knowledge of XML syntax, and basic experience in the creation of abstract workflows.

The applicability of the Transit Model is proved by introducing the Transit Model Solution, an open source compound transaction management system based on the Transit Model. Secondary objectives of this project include the experimentation with advanced transaction handling features, such as the introduction of long running transaction suspension and resumption mechanisms into the transaction management engine, together with the introduction of novel architectural concepts, such as the separation of transaction management system design from model definitions, thus creating a "pluggable architecture" where a management system can switch transaction models. This makes the system suitable to a wider range of applications.

Project Methodology

While this project is heavily research oriented, the formal methodology used in the development of the Transit Meta Model and Solution has been based on a hybrid of spiral and evolutionary prototyping. The general architecture of the solution is displayed below:



```
<?xml version="1.0"
encoding="utf-8" ?>

<model>
    <name>LLT Model</name>
    <decl>
        <activityList size = "*n*">
            arrayOfActivities
        </activityList>
    </decl>

    <workflow>
        <segment id = "Start">
    ...
    ...
    ...
    ...
```

The transit model solution has been designed in the form of an API, in order to be easily integratable into top level solutions. In the example above, a Transit Enabled Travel Agent System's architecture can be observed. The transaction model used by the Travel Agent's System is defined in the Transit Script File, while the Transit Model API provides necessary facilities for the creation of a long running transaction, which is then executed according to the model currently "plugged" into the Transaction Manager.

Results and Achievements

Various examples and practical scenarios have been tested using the Transit API, and models defined using the Transit Meta Model, including Travel agent and real live Electronic Account Top up scenarios. It can be concluded that the issues identified in the research phase of this project have all been addressed and resolved in one way or another, thus making the project a successful one.