10. Future Directions

As mentioned earlier there is a lot of research work on this area. There are numerous directions in which this project can be extended. These are discussed in this section.

10.1 More Cross cutting concerns

The robustness and quality of the applications generated solely depend on the framework support for the cross cutting concerns. In other words cross cutting concerns should be "built-in" to the framework. Issues like transaction handling, concurrency etc. already handled in the system. However the following important issues were scoped out of the project:

- Security: The generated web applications could be made resistant to common web threats like SQL injection, URL hijacking etc.

- Authentication and Authorization: The web applications could be provided with Loin and Logout facilities with the handling of users and roles. However the integration of this into the domain level modeling if needed would be a challenge.

- Logging: The web applications generated could be made to provide important logging information.

- Exception Handling: Exception handling has been built into the system to a certain degree with the use of AJAX frameworks. However this can be extended to include general application exceptions.

10.2 More modeling scenarios

The project has only scoped on a limited set of modeling artifacts. This could be expanded to cover more. Following are some of them which would add more value.

10.2.1 State Chart Diagrams

The state chart diagrams for each domain entity could be transformed into a finite-state-automata implementation. The current state should be persisted. The transformed business logic would tell which events are possible from the current state. These events can be provided as menu options in the UI, where clicking on such a menu option would perform the state transition.

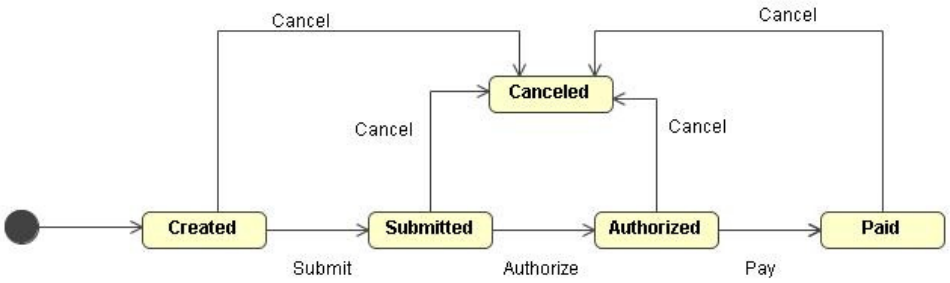

Figure 10-1 Sample state chart diagram

The above is a sample state chart diagram of the Order domain entity. If the current order is on say Submitted state, two menu options 'Cancel' and 'Authorize' should be enabled. Now if the user clicks on 'Authorize' the domain entity's state should be changed to 'Authorized' and now 'Cancel' and 'Pay' should be enabled.

The above is a common business requirement and would add great value to the system.

10.2.2 Enumeration type classes

Enumerations are common in modeling business domains. If a particular domain entity has a property for which values are from an enumeration, that enumeration could be made an enumeration class and made an aggregate by value relationship to it. The values could be made available as a combo box in the UI.

10.2.3 Special handling for inheritance hierarchies

Handling of inheritance is a much discussed area in ORM literature. The basic problem is that in the object oriented world inheritance is one of the most fundamental and integral

notions. However the same is absent in the relational world. Therefore there are few ways in which inheritance hierarchies are handled.

- Single table per hierarchy
- Table for each class in the hierarchy
- Table for each concrete class in the hierarchy

The current implementation identifies each domain entity in an inheritance hierarchy as unique and thus falls into the second category. However the first choice has few optimizations from the database side. Most ORM tools provide these facilities where the optimizations could be done "under the hood".

10.3More features

There are few features which could add so much value to the generated application. Some of them are discussed here.

10.3.1 Domain Entity Search

A search facility for each domain entity could be provided. The searchable attributes could be marked in the model with a predefined stereotype like say <searchable> and those fields would be available when searching for records of that domain entity.

10.3.2 Full AJAX support

AJAX has only been used for the validations in this project. However the AJAX framework used does provide components where the whole application could have AJAX functionalities. For example the table component provides client side sorting which is very useful in a business application.

10.3.3 Application Search

A very useful although challenging requirement would be to provide application wide search, something like Google inside the application.

10.4Decoupling from technologies

The project has used a lot of technologies. However each technology in the process should ideally be replaceable with another similar technology. For example Hibernate should ideally be replaceable by "Apache iBATIS". This ultimately boils down to the flexibility of the transformation. That in the transformation could be made to accept the technologies as transformation parameters. The framework which support the generated applications should also be pluggable to suite the different transformation scenarios.