Comparing the different mindsets between application developers and framework developers

After doing this project, I became more aware of different level of thinking that's needed to be carried out when working as an application developer and while working as a framework developer. While working as a framework developer, we have to think in a high level design approach so that the framework becomes flexible and reusable whoever uses it. And the first and foremost thing is that the non-changeable parts should go into framework and the changeable parts should go into the application. The framework level development demands for loosely coupled code most of the time. The decoupled code makes the side effects less, thus making the framework flexible without having dependencies among the modules. Module dependencies means the error propagation along the modules. Motives of most of the design patterns is to make the decoupled code. So, the usage of design patterns during framework development makes it the having decoupled code. During framework development, we have to think in terms of application developers' perspective as well so that the application developer can rely on framework for the cross cutting concerns and reusable aspects of software development like logger, security management, transaction management etc.

As an application developer, I need to be more focused on implementing the business logic and the realization of client requirements into the usable application for client. I used the reusable modules defined in the framework while developing an application. I used the command to encapsulate the request as an object which is taken care of by the framework. Similarly, I used the different types of logging strategies that has been provided by the decorator pattern implementation in framework. So, the development time was faster after using the framework for application development.

In this project, we used the patterns from creational, structural and behavioral, for example, singleton, factory method, proxy, decorator, command, strategy, template, iterator, façade, bridge patterns. We tried to separate out the concerns like transaction management, logger and used command to execute and undo the transaction related requests. I was more focused on developing the creation/updating of an application user, checking in the products. I used command pattern in handling the check-in request which further delegates the responsibility to façade for separating out the client from different interfaces under the hood. The command provides the undo operation as well in case of any failure of request handling. If any failure in database operations occurs, the command pattern is used to simply undo that request. I developed the template pattern to provide the similar steps of operation during check-in and checkout of the product, for example, saving the checked out/checked in products into database using command pattern, sending notification as either email or SMS using strategy pattern and then simply printing the bill of checkout/check-in. I implemented the façade pattern to define a high level interface that hides the complexity of subsystems for database operation.

So, in conclusion, as commonly said, the framework is a semi completed application handling the reusable and cross cutting concerns along with performance and optimization. While an application developer is more focused on realizing the client requirements into a complete application usable by client and may use the framework to make software maintainable, flexible, reusable and efficient.