The following is a review of our analysis phase documentation for the CashStash financial management application.

The domain analysis portion of the documentation begins with the concept statement, which outlines the application’s purpose, features, and intended users. By adapting the conceptual classes described in the concept statement, a conceptual domain model was developed, illustrating the attributes of the conceptual classes—including the actors involved in the application, the different objects handled by the application, and even external systems the application interacts with—as well as the classes’ relationships (IE parent-child) and interactions with each other. The process of thinking through the development of this model proved useful in conceptualizing the overall structure of the application and its interactions with the user and other agents.

Along with these portions of the domain analysis, a number of OCL constraints were included in the documentation; these constraints describe rules pertaining to possible values for the attributes of conceptual classes described in the concept statement and domain model. Specifically, the provided constraints state that the net dollar value of a transaction must be nonzero and that various external financial accounts must be verified via authenticator for use in the application. While keeping these constraints in consideration while designing the application can be useful, having more formally-defined constraints than just these would likely aid even further in the design process.

The application analysis portion of the documentation, which focusses on the internal functionality of the application, begins with the application interaction model, which describes a number of use cases (fifteen, to be precise). An essential use case describes the interactions between the user, the application, and any external systems the application interacts with, as well as error/exception handling. The concrete use cases for the application describe in detail specific ways in which a user would use the application and highlights the interactions between the user and other actors, objects, and external elements for each given case, as well as the preconditions for each case and how errors/exceptions in each case would be handled. From both the essential and concrete use case descriptions, scenarios can be written demonstrating how the execution of each case would proceed using a third person example (“John Doe” scenarios). From the essential use case and its scenario, the high-level system sequence diagram (HSSD) is drawn; from the concrete use cases and their scenarios, the detailed system sequence diagram (DSSD) is drawn. Both system sequence diagrams display the use cases graphically, describing the actors involved, the interactions between them, and any internal events. The DSSD takes this further by identifying boundary objects, which represent the interfaces through which users interact with the system, and controllers, which handles internal processes and may request interaction from the user.

Based on the use case scenarios outlined in the DSSD, the application class diagram highlights the various classes of objects (as seen in the DSSD) and their attributes and associations. Also based on the DSSD, especially the controllers, the application state diagram highlights the various states of the application and the transitional events (interactions with the controller) involved in moving between these states. Both of these graphs are invaluable to the visualization of the application as a whole, how it and its classes structured, and how it might operate.

Finally, the conceptual domain model and the application class diagram are combined into the consolidated class model, with the goal of visualizing the detailed processes of the application in the context of their place within the generalized structure of the system.