**Prototype Justification**

This team chose to iteratively develop the ViKER software package – back-end and front-end – according to an evolutionary, horizontal model. Horizontal prototyping refers to taking a broad approach to developing a system. That is, user interaction with the system is deemed more important than the specifics of the implementation of the system.

The main goals of this project were outlined during phase 1 of this development cycle within the scope documentation. These goals were given as follows:

1. Implement transformations from EER to ARM and from ARM to EER as described in the KnowID paper provided by the client.
2. Report on success/failure of a transformation.
3. Report on those things that could not be transformed (error reporting).
4. Report on what happened with each element.
5. Build a “point-and-click" GUI to allow users to construct the query visually.
6. Implement save/load functionality for models.
7. Construct at least 10 test cases.

The ends of all these goals result in visual feedback to an end-user. It is clear that how a user interacts with the system is critical to the success of this project.

It was thus an actively decided to take an evolutionary approach to development of this project. With an evolutionary approach both the team and client can grapple with the deliverables throughout the process of development. What is possible can be determined by the team early on the process, and the specific design steps taken throughout the project can reflect this.

Furthermore, the prototype is horizontal as there are several distinct aspects of this project which need to be developed. It was deemed most appropriate to develop these aspects simultaneously since the different aspects of the project will influence the other aspects of the project. Take, for example, the user interface. The UI must display and provide feedback to users based on their interaction with the system. Designing the UI to display diagrams (ER or ARM) is dependent on how the back-end system interprets the diagrams themselves.

The UI is a split screen view with the input on the left and the output on the right. We chose to display the models this way as it shows a clear transformation from one model to the other. We decided to use React JS a front-end framework to develop the UI as it has access to great diagramming libraries. React renders components with HTML and CSS which makes styling the UI a lot easier to implement.