An Architects Story that convert Project Failure to a Production Success.  
This is a true story of how one project destine to fail was turned into a Successful Deploy to Production with Zero Defect.

Written by Carl Boudreau, Software Architect for Modern Clinical Delivery  
carl\_boudreau@optum.com

*If you haven’t read my article “****Start with a Wireframe! Agile is not an excuse to start coding without a plan****” I would recommend you read it and save you and your team a lot of frustrations while developing your application.*

At the onset of Project X there was a lack of technical expertise & leadership without the ability to have a Big Picture prediction. It was a flawed thought; to assume that because they are following agile they can just start driving and eventually get to where they wanted to be. Without knowing the Big Picture or having the experience to know what technical challenges lies ahead in the Big Picture, one is bound to drive in the wrong direction, or into a dead end.

The first 3 months of the Project X had no demonstrable progress. People didn’t have any experience in writing software “did not know what they didn’t know” so the wrong resources were picked, the wrong technologies were picked, and wrong priorities were set, to make matters worse there was no experienced technical leadership to provide direction.

Once a resource was added that had the technical experience & leadership to choose the best Technology Stack and Architectural Design, along with the right resources with the right skill sets Project X was set to succeed.

***Easily written than done***

***Big Picture***  
To describe a software solution into an agile tool like Team Foundation Server or CA Rally one needs to know how to disassemble an object. An understanding of the Big Picture (feature) is required. One needs the ability to reverse engine and disassemble the picture into small digestible components (user stories) in their mind along with the experience to understand the raw materials (technology stack), tools (resources with correct skill sets) and steps (tasks) needed to produce the Components.

With a list of Components needed, one needs the experience to know what Components are needed to be completed in a logical order (Feature, Story & Task Prioritization) to demonstrate progress. With the list now one can size and plan for a demonstration (Sprint.) Equally important is the build and assembly process.

***Build, Assembly and Design***  
As the industrial revolution has proven in history the process of Assembly is a Design that reduces complicated task into an organized structure. Proven in CPU design; parallel processing greatly speeds the process of performing tasks. Choosing an Architecture Design that combines Organized & Parallel Assembly helped to separate Look & Feel, from Presentation Layer, from Business Logic, from Data Storage. This Separation allows multiple works to be completed in parallel, and then assembled in a modular fashion.

***Modularization***  
Because an Architectural Design was chosen, the team was able to work on the three major aspects of producing a successful software package. The design separated Look & Feel, from Presentation Layer, from Business Logic & Data Storage (API). They gain 3 times the speed of a single developer in a standard software delivery cycle. One team worked on the API, another worked on the Presentation layer and another worked on Look & Feel. Within the Presentation Team and API team they were able to benefit from parallel efforts to produce Extensions. An Extension is a packaged functionality that works with other extensions to provide the larger program functionality. Extensions can be removed, upgraded and replaced without disrupting any other team or Developers, Testers and UAT teams. In addition the Business can decide when; where and who sees the Extension for how long using publishing features in the application.

***Final Production Result***  
The final result was happy customers and zero defects  


***Conclusion***  
Choose a modularized framework that is extendable. Reverse engineer the Big Picture to understand how it can be broken into smaller parts. Create a pattern to reassemble those smaller parts in a manner that has no dependencies or blockers that would prevent building it a parallel fashion. Set priorities based on technical need and then business need.