# Reduce Critical Engineering Dependencies

August 2016

With software eating the world, your job has never been more demanding. Part of that includes getting application functional requirements and the technical architectures to your development teams so they can deliver fully functional prototypes. This process takes quite a bit of time.

In this read, we will discuss how you can create fully functional prototypes without depending on critical engineering resources, and get to market very fast.

Metavine, Inc.

2001 Gateway Place
Suite 330E
San Jose, CA 95110

© Metavine, Inc.

www.metavine.com

# Reduce Critical Engineering Dependencies

As a solution architect or business system analyst, you need to go to your technical resources to get certain things done. Whether it's connecting to a system of record, creating a prototype, extending a system solution, or adding a new field to a data table, your technical resources are imperative and necessary for your success. Traditional software development and architecture processes, however, require a great deal of technical engineering resource. Today's agile software methods are designed to help address this but do not eliminate the need for those technical resources. What's required is an agile model that lessens your dependancy on scarce and critical engineering resources and allows you to create functional prototypes that match your exact requirements. If you can mitigate your engineering dependency, then you can deliver product to market very rapidly.

A Functional Design Document (FRD) is used to create specific application or architecture requirements. There are four elements that defines the overall UX experience: data, workflow (processees), rules (logic) and UI. For this paper's purpose, we will use the idea of creating a functional prototype as our anchor. **1) Data:** This is where structured and unstructured data are discussed as well as definitions, integrations and ontologies. At this stage, people with database skills are most usually required to help establish the required data models. Newer software, however, allows non-DBA types to structure their data models without requiring technical skills. These Next-Gen solutions are smart enough to remove the complexity and put data definitions and integrations into the background, thus removing complexity. **2) Workflow:** Here is where a great deal of pain exists between design and development, or between the business unit and engineering team. It is hard to verbally write out what one wants a workflow or process to be. Next-Gen solutions help by having you create a visual workflow, akin to a visio diagram, that makes the data paths and the user experience clear and unambiguious. **3) Rules:** A rule, or logic, is something that applies to both data and experience. In essence, what behaviors and actions do you want to take against a given data set or experience. Next-Gen solutions use natural language terms, not programming terms. When you use natural language, then everyone can understand, without ambiguity. **4) UI:** UI design normally requires HTML, PHP or CSS technical skills to build user facing forms. Next-Gen platforms make it easy to create these forms, giving you the look and feel your users require with exacting specificity. This is important for user acceptance and adoption.

Our Next-Gen Platform is comprehensive across each of these four steps. Requirements are specified in the Metavine Design Studio, incorporating a holistic, zero code approach to the Data, Process, Logic, and UI layers, allowing for rapid prototyping. Once one specifies a solution in Metavine, a functioning solution results and is ready for user acceptance testing and the design iterations to follow. This means that the right prototype is delivered to market exceptionally fast; up to 10x faster than today's norms.

2001 Gateway Place
Suite 330E
San Jose, CA 95110

www.metavine.com

For more information
email us at
info@metavine.com