**What are some**[**modelling**](http://mydy.dypatil.edu/rait/mod/forum/view.php?id=66976)**strategies that can be used when a system is too big or complex to create a complete architectural model?**

A common agile practice is to perform some high-level architectural modeling early in the project to help foster agreement regarding your technical strategy within the team and with critical stakeholders. The goal at this point is to identify an architectural strategy, not write mounds of documentation, enabling you to do this swiftly.

[Agile Model Driven Development (AMDD)](http://agilemodeling.com/essays/amdd.htm), see [Figure 1](http://agilemodeling.com/essays/initialArchitectureModeling.htm#Figure1), explicitly includes an initial architectural modeling effort during [Iteration 0](http://www.ambysoft.com/essays/agileLifecycle.html#Cycle0) of an agile project (what some processes might call the Warm-UP, Inception, or Initiation phase). Initial architecture modeling is particularly important for scaling agile software development techniques to large, complex, or globally distributed development (GDD) efforts.

Some people will tell you that you don't need to do any initial architecture modeling at all. However, my experience is that doing some initial architectural modeling in an agile manner offers several benefits:

|  |
| --- |
| 1. Improved productivity. You can think through some of the critical technical issues facing your project and potentially avoid going down fruitless technical paths. 2. Reduced technical risk. Your team gains the advantage of having a guiding vision without the disadvantage of having to overbuild your system – just because you've modeled it doesn't mean you have to build it. 3. Reduced development time. Initial agile architecture modeling enables you to make better cost and time estimates for your project, two pieces of information which management will want. 4. Improved communication. Having a high-level architecture model helps you to communicate what you think you're going to build and how you think that you'll build it, two more critical pieces of information desired by management. 5. Scaling agile software development. Your initial architecture model will be a key work product in any "agile at scale" efforts because it provides the technical direction required by sub-teams to define and guide their efforts within the overall project. 6. Improved team organization. Effective teams are organized around the architecture or line of business, not around job function. As you scale to larger and/or distributed teams the sub-teams should each be responsible for one or more sub-systems -- you don't want to organize your sub-teams around job function (e.g. an architecture team, a development team, a testing team, ...) because that requires you to increase the documentation and bureaucracy overhead which in turn increases risk, cost, and time to value. |

Early in the project you need to have at least a general idea of how you're going to build the system. Is it a mainframe COBOL application? A .Net application? J2EE? Something else? To do this the developers on the project will get together in a room, often around a whiteboard, discuss and then sketch out a potential architecture for the system. This modeling work is based on, and performed in parallel to, your [initial high-level requirements modeling](http://agilemodeling.com/essays/amdd.htm#InitialRequirementsModeling) efforts. Your architecture will evolve over time so it does not need to be very detailed yet (it just needs to be [good enough](http://agilemodeling.com/essays/barelyGoodEnough.html) for now), and very little[documentation](http://agilemodeling.com/essays/agileDocumentation.htm) (if any) needs to be written.

Top of Form