ThoughtWorks®
STUDIOS

The Agile Maturity Model
Applied to Building and Releasing Software

By Jez Humble and Rolf Russell | September 2009

In this paper, we present a maturity model for building and releasing software.
This model is designed to serve several purposes:

- To provide a structure for assessing your team or organizational capabilities in building and releasing software
- To provide an approach for planning and executing improvements to existing practices

We start with a discussion of the Agile Maturity Model, move on to building and releasing software, present the maturity model, and then describe how to use it.

The Agile Maturity Model

The Capability Maturity Model Integrated (CMMI®) is intended to institutionalize a collection of pre-defined delivery practices and ensure their consistent execution so as to increase the probability that a team or organization can successfully complete projects. The definition of "successful" includes completing the project on time and in budget.

In contrast, the Agile Maturity Model is an internal tool used at ThoughtWorks and other organizations to help organizations understand their current practices and work toward improving them with the goal of increasing ability to respond to changing business conditions and better harnessing innovation.

The model used here is both a specialization and an adaptation of the Agile Maturity Model. Although we share the same goals as the Agile Maturity Model, we have changed the definition of the levels, so as to apply it to the practices related to building and releasing software.

Building and Releasing Software

The delivery of working software involves several activities besides development. In particular, software must be turned into a form suitable for installation in its target environment, subjected to various forms of testing, and then be released to users.

Our assertion is that this process should be performed continuously, rather than as a series of phases, with the aim of creating a fully automated, reliable, predictable, and visible process with well-understood, quantifiable risk.

The Ideal State

Ideally, each change made to your application, its environment, or its configuration should go through an automated process. This process should create binaries, run automated tests against them, and inform all relevant parties of the results of this process. We call such a system the deployment pipeline.

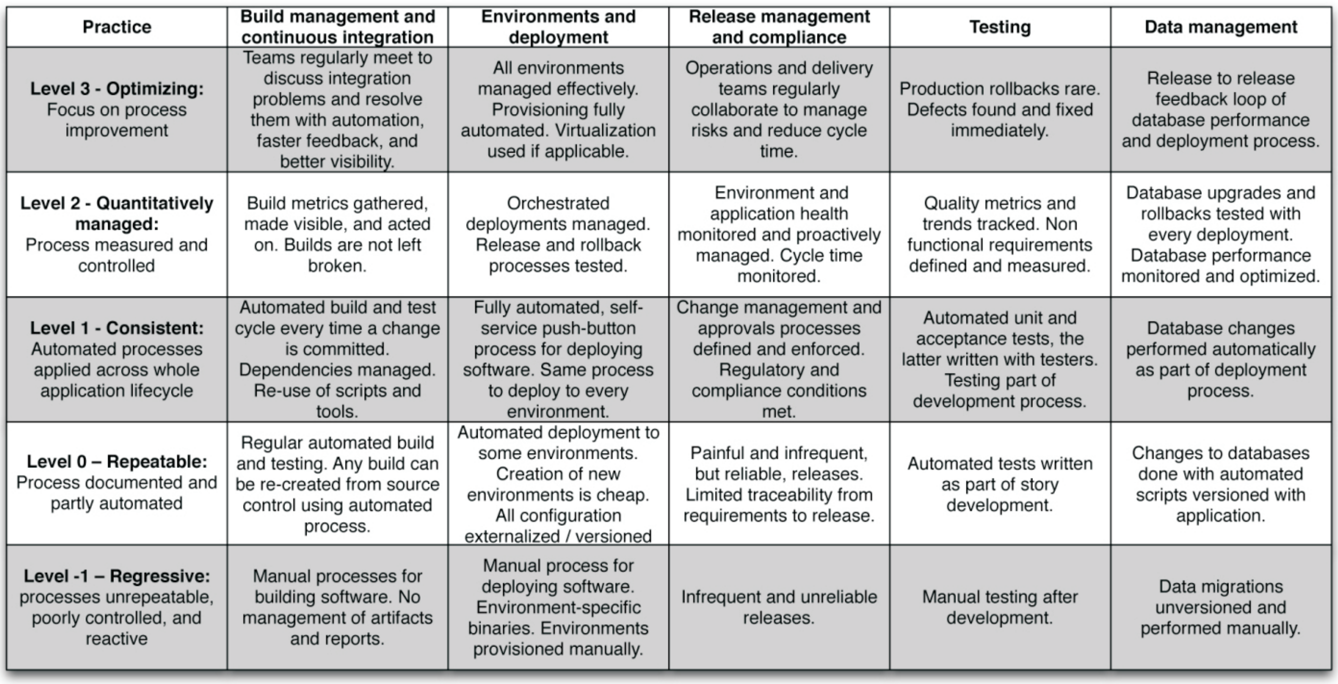
It should also be possible for developers, testers, and operations people to have not only visibility into this process, but also to be able to self-service processes such as deployments to testing and environments and the release of applications at the push of a button. Such processes also form a part of the deployment pipeline.

The Maturity Model

In order to achieve our ideal, it is essential to cover all parts of the process of building, deploying, testing, and releasing software.

> Build management and continuous integration are concerned with creating and maintaining an automated process that builds your application and runs tests on every change and then provides feedback to the whole team on the process.
>
> Environments consist of the entire stack your application requires to work:
> hardware, infrastructure, networking, application stacks, external services, and their configuration.
>
> Release management is defined by Forrester as “the definition, support, and enforcement of processes for preparing software for deployment to production.” We have added considerations around compliance to this area, since conformance to regulatory environments is often one of the strongest constraints on release management.
>
> Testing, whether through automated tests or manual processes such as exploratory testing and user acceptance testing is designed to ensure that software contains as few defects as possible as well as conforms to non-functional requirements. We have focused on the areas of testing that are most relevant to building and releasing software.
>
> Finally, data management (usually, but not always, in the context of relational databases) forms an essential part of the deployment and release process, since it is a frequent source of problems when releasing or upgrading software.

To ensure each part of the process is given due attention, we have divided the model into five sections.



Practice	Build management and continuous integration	Environments and deployment	Release management and compliance	Testing	Data management
Level 3 - Optimizing: Focus on process improvement	Teams regularly meet to discuss integration problems and resolve them with automation, faster feedback, and better visibility.	All environments managed effectively. Provisioning fully automated. Virtualization used if applicable.	Operations and delivery teams regularly collaborate to manage risks and reduce cycle time.	Production rollbacks rare. Defects found and fixed immediately.	Release to release feedback loop of database performance and deployment process.
Level 2 - Quantitatively managed: Process measured and controlled	Build metrics gathered, made visible, and acted on. Builds are not left broken.	Orchestrated deployments managed. Release and rollback processes tested.	Environment and application health monitored and proactively managed. Cycle time monitored.	Quality metrics and trends tracked. Non functional requirements defined and measured.	Database upgrades and rollbacks tested with every deployment. Database performance monitored and optimized.
Level 1 - Consistent: Automated processes applied across whole application lifecycle	Automated build and test cycle every time a change is committed. Dependencies managed. Re-use of scripts and tools.	Fully automated, self-service push-button process for deploying software. Same process to deploy to every environment.	Change management and approvals processes defined and enforced. Regulatory and compliance conditions met.	Automated unit and acceptance tests, the latter written with testers. Testing part of development process.	Database changes performed automatically as part of deployment process.
Level 0 – Repeatable: Process documented and partly automated	Regular automated build and testing. Any build can be re-created from source control using automated process.	Automated deployment to some environments. Creation of new environments is cheap. All configuration externalized / versioned	Painful and infrequent, but reliable, releases. Limited traceability from requirements to release.	Automated tests written as part of story development.	Changes to databases done with automated scripts versioned with application.
Level -1 – Regressive: processes unrepeatable, poorly controlled, and reactive	Manual processes for building software. No management of artifacts and reports.	Manual process for deploying software. Environment-specific binaries. Environments provisioned manually.	Infrequent and unreliable releases.	Manual testing after development.	Data migrations unversioned and performed manually.

How to Use The Maturity Model

The ultimate aim for your organization is to improve. The outcomes you want are:

- Reduced cycle time, so you can respond faster to changing business requirements and increase revenue
- Reduced defects, so you can improve your reputation and spend less on support
- Increased predictability of your software delivery lifecycle to make planning more effective
- The ability to adopt and maintain an attitude of compliance to any regulatory regime you are subject to
- The ability to determine and manage software delivery risks effectively
- Reduced costs due to better risk management and fewer issues delivering software

We believe that using this maturity model can help you achieve all of these outcomes.

Here are the steps to use the model, based on the Deming cycle: plan, do, check, act[1].

1. Identify where your organization lies on the model. You may find that different parts of your organization achieve different levels on each of the different categories.

2. Choose what to focus on. You should work out what the possible improvements you could implement are, how much they will cost, and what benefit they will deliver. You then choose a few improvements you could make and decide how to implement those changes. You should set acceptance criteria to define the results you are expecting to see, so you can decide if the changes were successful.

3. Implement the changes. First, plan how to implement the changes. You might decide to start with a proof of concept. If so, choose a part of your organization that is really suffering—these people will have the best motivation to implement change, and it is there that you will see the most dramatic change. Then, of course, you have to execute your plan.

4. Check if the changes you implemented had the desired effect. Use the acceptance criteria you created. Hold a retrospective to find out how well the changes were executed.

5. Repeat these steps, building upon your knowledge. Roll out more improvements incrementally, and roll them out across your whole organization.

Organizational change is hard, and a detailed guide is beyond the scope of this paper. The most important advice we can offer is to implement change incrementally. If you try and go from level one to level five across your whole organization in one step, you will fail. Changing large organizations can take several years.

Finding the changes that will deliver the most value and working out how to execute them should be treated scientifically: come up with a hypothesis and then test it. Repeat and learn in the process. No matter how good you are, it is always possible to improve. If something doesn't work, don't abandon the process: try something else.

ThoughtWorks® STUDIOS

About ThoughtWorks Studios

ThoughtWorks Studios is a global leader in Agile software development tools, and its products can be found in development organizations seeking sustainable Agile adoption. The company's Adaptive Application Lifecycle Management (ALM) solution provides a platform for managing all aspects of software development, from requirements definition and project management to test automation, quality assurance, and release management. Adaptive ALM comprises the integration of three products: Mingle (Agile project management), Twist (Agile testing), and Go (Agile release management). Each tool is available as part of a complete lifecycle solution or as a standalone product. Backed by more than 17 years of experience in Agile delivery, ThoughtWorks Studios is the product division of ThoughtWorks Inc., a pioneer in Agile development. ThoughtWorks Studios has over 400 customers in more than 20 countries, including 3M, Honeywell, BBC, eBay, Barclays, Vodafone, McGraw-Hill, and Rackspace. The company headquarters is located in San Francisco and Bangalore, with offices in London and select cities in Europe, Asia, and Australia. For more information, visit **www.thoughtworks-studios.com.**

mingle®

Mingle, an Agile management and collaboration tool, provides a common workspace for all team members and an automated system of record for all projects. Mingle can adapt any existing workflow process and easily manages daily development activities. Offering true-to-life visibility in the entire development process for all stakeholders, Mingle helps development teams become more open and collaborative.

go™

Go is a solution for Agile release management, which enables businesses to release software on demand. Go improves collaboration between developers, testers, and operations and provides fast feedback on the production readiness of your software. Using Go, teams can model the delivery process, perform push-button deployments, and trace from deployments back to version control.

twist®

Twist, an automated Agile testing solution, provides English-like constructs, making the testing process more productive for all team members. As applications grow in complexity, Twist helps to more easily maintain complex test suites. These suites keep pace with application development and are held as long-living assets.

ThoughtWorks Studios

CALL: 512-467-4956 (sales)
415-238-6497 (main) North America

EMAIL: studios@thoughtworks.com
WEB: www.thoughtworks-studios.com

+91 80-4064-9703 | Rest of the world

SAN FRANCISCO | CHICAGO | LONDON | BANGALORE | BEIJING | MELBOURNE