Reality Checking Agile’s Architectural Inner Workings
By demystifying Agile constructs, and how architects fit into the development process, organizations can find and follow best practices and deliver benefits that advance accelerated coding objectives and meet strategic business needs.

Executive Summary
Although Agile has been around since 2001, some organizations still find the concept of “Agile architecture” confusing. On one hand, they hear “don’t do up-front design” – but this is often counteracted by the fact that if an organization does not design early, it will face major challenges in proceeding with software visioning and/or framework development.

Agile philosophy warns organizations to not attempt “big design up front” because doing so may result in teams scraping a major chunk of hard work if the business decides to change its approach. Such a change can arise from any one of numerous reasons: changed market requirements, staying ahead of the competition, etc. And this dilemma forces Agile advocates to ponder, how much architecture is just right in Agile? Although there may be no absolute answer to this quandary, this paper will explore how an Agile team can find the best balance of up-front architecture work.

Assessing Architecture Requirements
Honestly, there is no single, fixed answer to the question of how much architecture is just enough. The answer depends on the project’s context and team members. Ultimately, it is up to the team to figure out how much to architect up front – by applying their collective wisdom. However, in general, the idea is to not spend much time designing and implementing various moving parts – hinged-to layers and tiers, with interdependencies or responsibilities and cross-cutting concerns. Rather, it makes more sense to simply build the minimal amount of code needed to connect all of the basic pieces (some of which may still be in the conceptual state) and start building the actual functionality on top – thus providing an early end-to-end experience of the results.

So the focus is more on the API level of the infrastructure and not the actual implementation – which is usually mocked up for the first few Sprints. And as iterations progress, the actual implementation is incrementally completed, as guided by the need of the other functional parts of the application.

Here is a guideline on getting started with an Agile architecture, distilled to seven simple steps:

- **Identify business objectives**: Focus on understanding “why” the business wants to build this, rather than “what” the business is planning.
Agile Architecture: A Sequential Process for Getting Started

- **Establish architecture objectives**: Identify technical challenges, understand/identify the technology stack, define major hardware/software requirements, agree on design patterns, identify component/service reuse, create high-level diagrams and define quality/security/performance attributes.

- **Identify major flows**: Draw the major flows — those that impact business operation, key scenarios/use cases, etc. — that help in validating the approach.

- **Draw an application overview (using a simple wireframe/screen mock-up/prototype)**: As a team, use the white board to draw the application overview, communications behavior, dependencies and layers/tiers.

- **Identify design coupling points**: While driving the design, you need to carefully identify the couplings so enough room remains for future realigning.

- **Create architectural candidate solutions**: Come to an agreement as a team and extract an architectural candidate solution. Make it transparent and open for criticism, and then fine-tune it further based on feedback.

- **Inspect and improve**: Once your candidate is ready, start rolling the development and continue improving it as you progress.

Organizations need to understand that Agile development starts even before the outcome is fully understood or envisioned. Design and development approaches are adjusted as development progresses, and teams use empirical data and knowledge to course-correct the architecture. So, architecture is a team effort. Also, instead of prescribing a specific approach, Agile architecture offers flexible solutions, thereby providing options in case one approach no longer serves the business’s altered needs. And in this process, we don’t have to design our architecture over a single iteration. Neither should organizations get lost in the details. They should start by setting their focus on the big steps and build a framework on which teams can base their architecture and design requirements on a preexisting foundation.

**Architecture Responsibilities Spelled Out**

Agile is all about team, with everyone on the team responsible for architecture. But there are two key drawbacks: People may have different opinions, and the approach may not scale (especially when we have a large, distributed team).

The solution is clear: Identify an architecture owner.

**Agile Architects**

Now the question for organizations is how to identify an architect owner. Should it be the traditional lead architect, or someone new to the world of Agile development?

Well, anyone who is experienced enough, skilled enough to drive the architecture and is sensible enough to understand business needs could be this person. However, be advised: This is indeed a challenging role. The product owner will require architectural counseling as and when he/she
creates a new vision. Developers are usually on their toes and may even glance at the architect with a heinous look. During Sprint planning, the team will need its architects to define the technology boundary of the work-items. The organization will expect architects not only to provide system architecture, but also to consider the big picture of enterprise architecture.

Ironically, if the organization looks at traditional architects moving into the Agile sphere, there is a palpable sense of resistance in the air. This resistance is attributed to the widespread myths of Agile architecture and its associated roles. These include:

- Sprint 0 is the time for a detailed architectural spec.
- Agile methods are architecturally weak, disconnected from the realities of delivering large systems in complex enterprise environments.
- Agile discourages “up-front” designing.

However, the facts suggest:

- During “Sprint 0,” teams need to get their project organized and moving in the right direction. But they don't need a detailed spec per se.
- Agile encourages waste reduction, as development is based on a moving target. But that doesn't mean Agile is architecturally weak. Rather, it helps us to be more aligned to realities and helps drive success by delivering complex enterprise projects.

Part of the up-front effort are initial requirements and architecture envisioning so that teams are able to answer critical questions about the scope, cost, schedule and technical strategy of their projects.

The question then becomes whether to engage in up-front design or not.

**Big Up-Front Designing vs. None**

Big up-front designing (BUFD) is on one extreme, whereas no up-front designing (NUFD) is the other pole. Agile architecture, however, is about some short small up-front design (SSSUFD). Keep in mind:

- Agile architecture is not a blueprint.
- Agile architecture is about what changes you can ignore.
- Agile architecture is about impactful choices up front that will make later choices easy.
- Agile architecture is subject to change.
- Agile architecture is about stable ground, but not frozen (static) ground.

So, how much rough up-front design (RUFD) is enough?

- It depends.
- Let it give you enough confidence to move forward, but don’t rush.
- It can take as long as two Sprints and be as short as “about a week.”

**Articulating Agile Architecture**

[Diagram of the Articulating Agile Architecture process]

**Figure 2**
• Deliver your architecture as code, as abstract base classes and some “base-lined” documents.
• It grounds you in reality and gives you the platform to take off and implement the user stories as you go forward.
• Spike it (i.e., create a technical debt item and drive it).

Agile is, and has always been, driven by moving targets (i.e., changing/evolving needs). Its architecture, however, has never wavered.

Evolving Architecture
Some thoughts to consider as your organization ponders Agile architecture:
• Architecture emerges over time, incrementally.
• Organizations need to work faster at first, due to the greater need to set the foundation of a project.
• However, the work evolves to reflect greater understanding and knowledge of the development team.

In addition, the following best practices should be followed once the foundational thinking is in place:
• Think about the future, but don’t overbuild the architecture.
• Be open about your architecture – encourage criticism, and allow your architecture to evolve.
• Use spikes; validate your model through implementation.
• Travel light – do not create a five-page document when a diagram will do.
• Allow requirements to drive your architecture updates.

About the Author
Arijit Sarbagna is a Senior Manager of Projects within Cognizant’s Advanced Solutions Group. He specializes in helping companies realize efficiency gains by adopting Agile principles and practices such as Scrum (and Extreme Programming). Arijit started his technology career in 1998 as a software developer, a development lead/manager, a business development manager and a variety of other jobs before settling on his true calling, project and program management in Agile space. Arijit is also the local Chapter Engagement Representative (CER) from PMI Agile Community of Practice (CoP) and represents the West Bengal, India chapter. He can be reached at Arijit.Sarbagna@cognizant.com.

About Cognizant
Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world’s leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 156,700 employees as of December 31, 2012, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world. Visit us online at www.cognizant.com or follow us on Twitter: Cognizant.