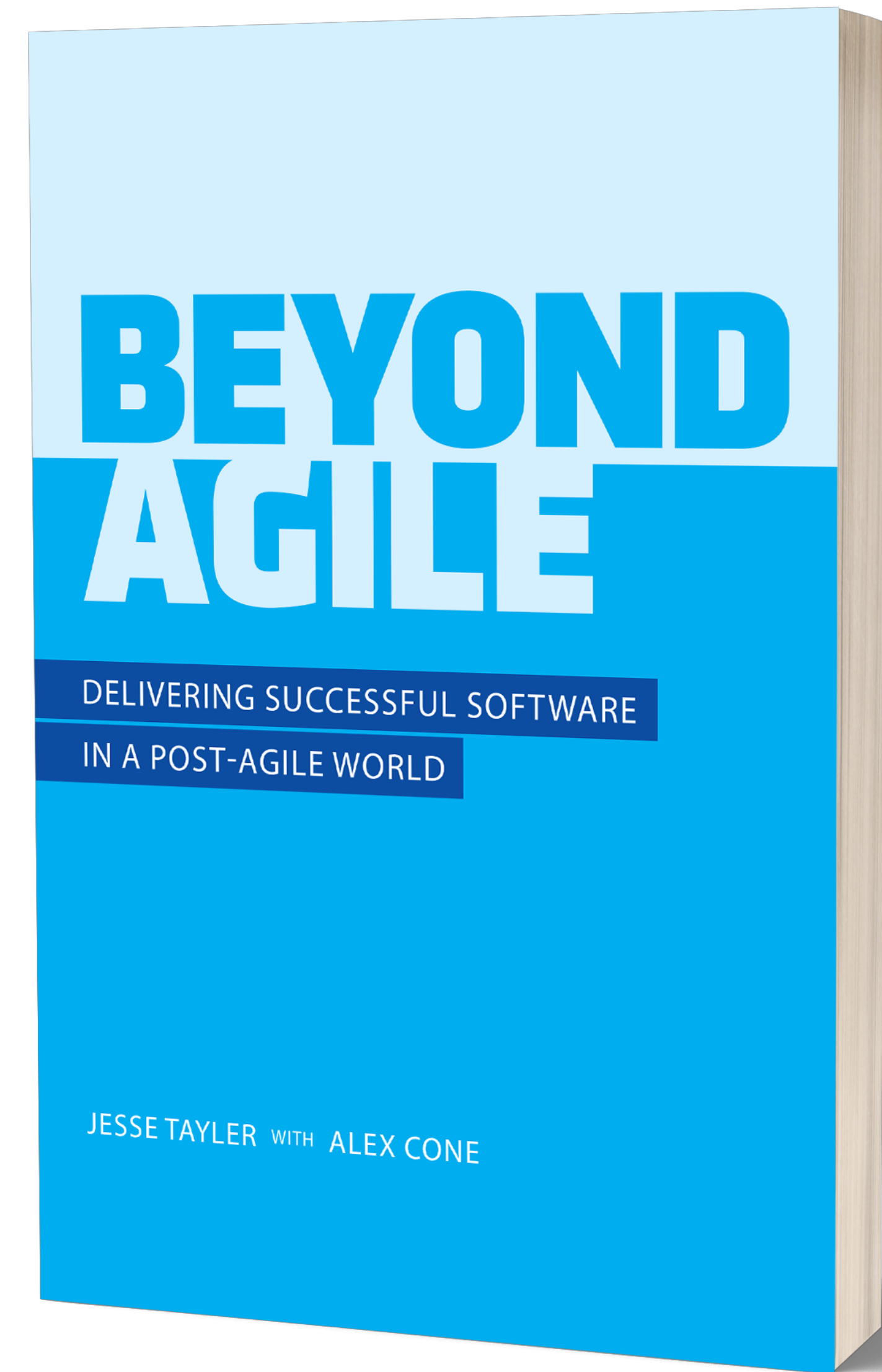


Achieve
excellence



Beyond Agile

By Jesse Tayler with Alex Cone



Master software architect, startup executive & App Store inventor.



Super-Geek, master-level Wall-St engineer & founder of CodeFab.

Process Follows Purpose

Process Follows Purpose



Cascade

Slow Reliable Process
Based On Pools Of
Human Computers
Crunching Numbers



Extreme

Small Teams, Pair
Programming
Emphasizes Design/
Invention



Agile

Continuous
Improvement, Brittle
Scripting Languages &
Fast Turnaround



Beyond Agile

Multi-Disciplines, Server
APIs, Adaptive
Interfaces & Mobile
Clients

“Design is a funny word. Some people think design means how it looks. But of course, if you dig deeper, it's really how it works.”

–Steve Jobs, American Entrepreneur

A low-angle, close-up shot of a person's feet wearing white bowling shoes with dark soles. The shoes are positioned on a polished wooden bowling lane. The background is blurred, showing the colorful lanes of a bowling alley. A semi-transparent white box with a thin border is centered over the image, containing the title and subtitle text.

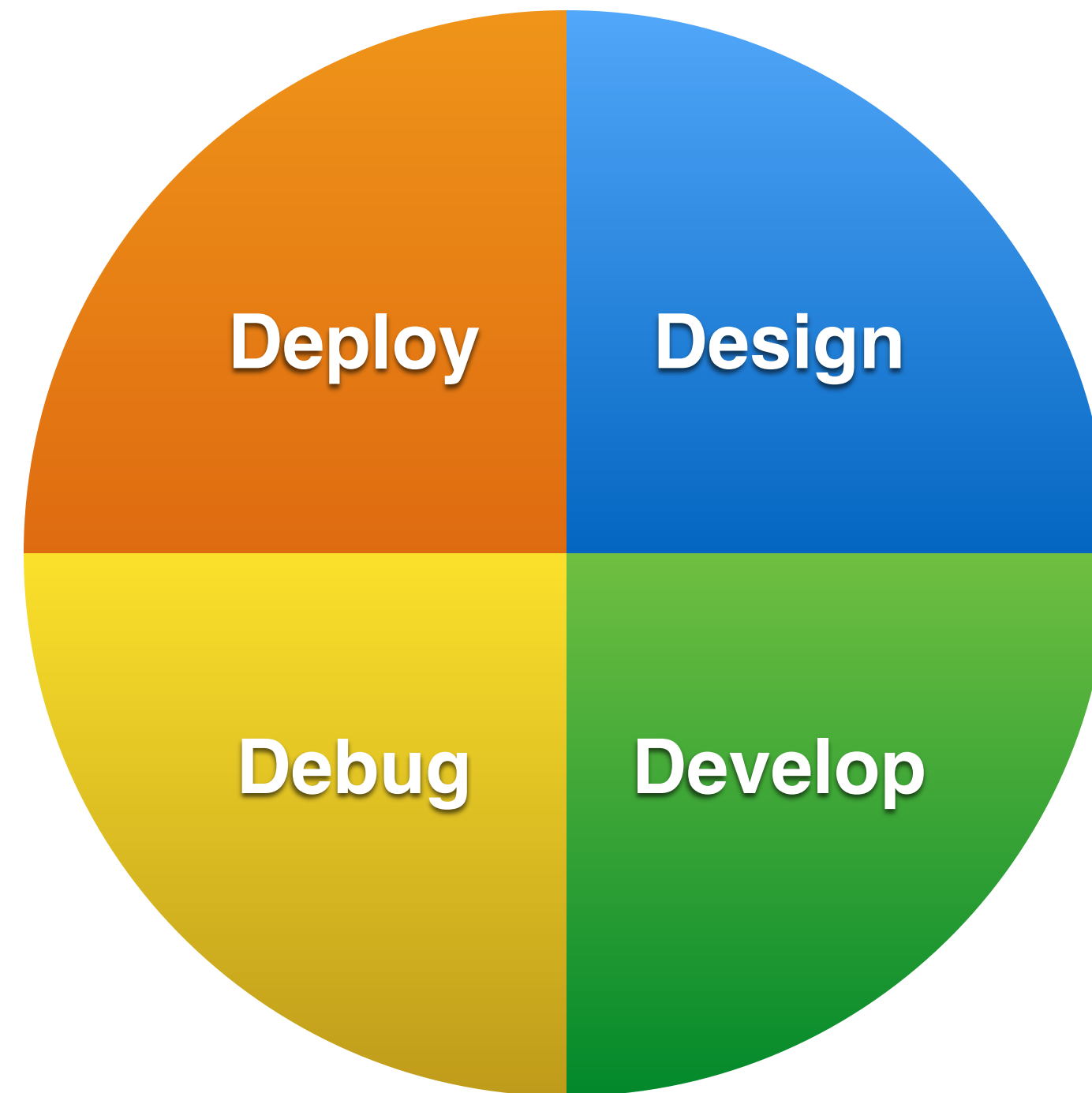
Bowling Has A Season

Why Would They Do That?

“The National Bowling League recognizes what many software project managers do not. It is the schedule itself, the cycle of seasons that provides the scaffolding for order, and the assurance of predictability”

–Beyond Agile

Phases



Seasons



Seasons...

natural changes in ***behavior*** and ***activities*** as we move through cycles

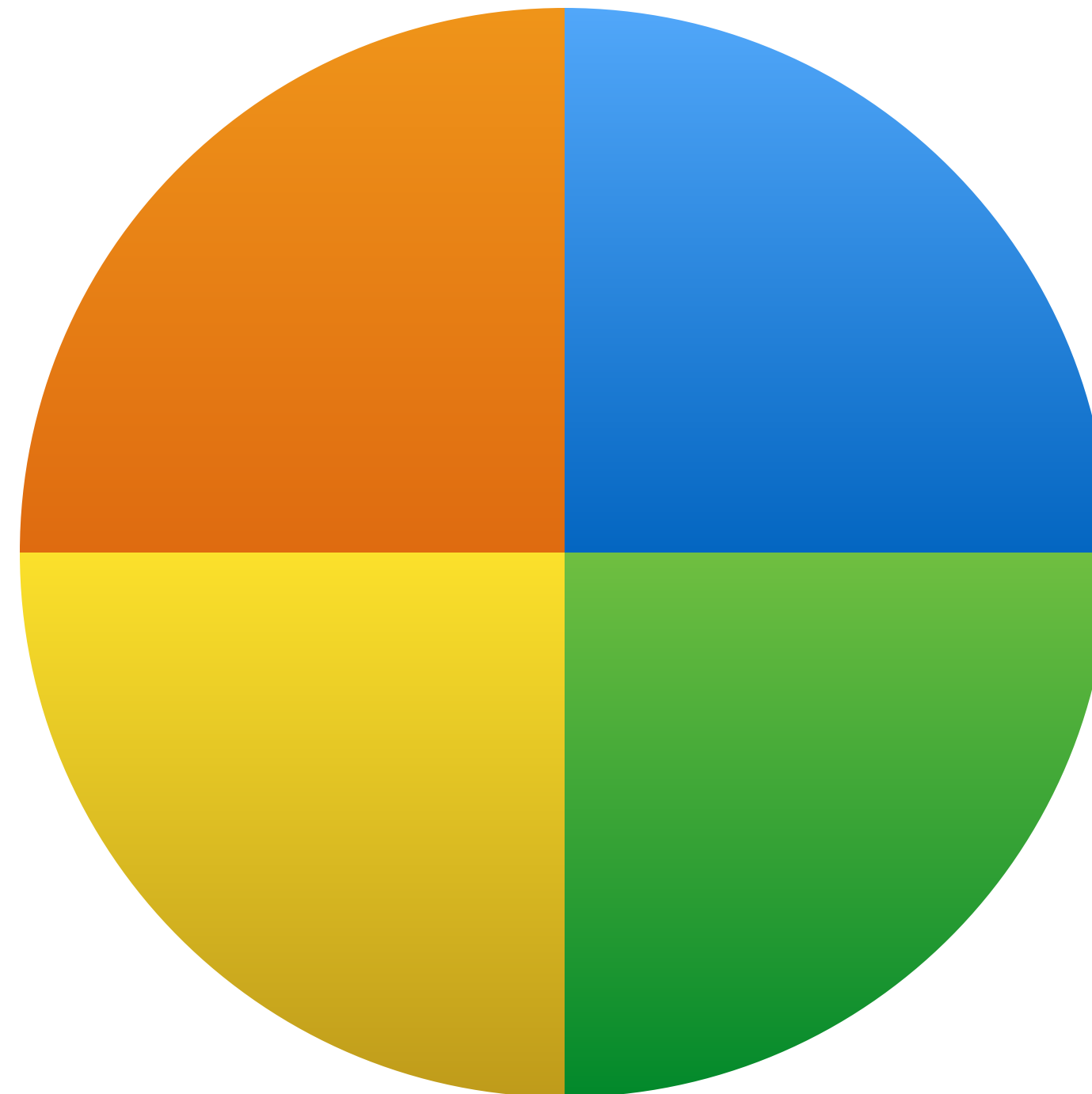
Phases

- **Design Freeform ideas**, sketches and research; attack hard or unknown issues. This is typically the ***most collaborative*** phase.
- **Develop Focus on the meat** of development, this phase is typically the longest and ***least directed*** phase.
- **Debug Cut-off unfinished work** & prepare for stable release. Hit lists of bugs, even named bugs if they are tenacious. This phase is often the ***most directed*** of any phase.
- **Deploy Procedures**: Controls & procedures. Follow the book. Countdowns, checkoff-lists and procedural guides predetermined for every step. Considered ***least collaborative*** because of studied and predicted situation handling.

Phases

Least Collaborative

Most Collaborative



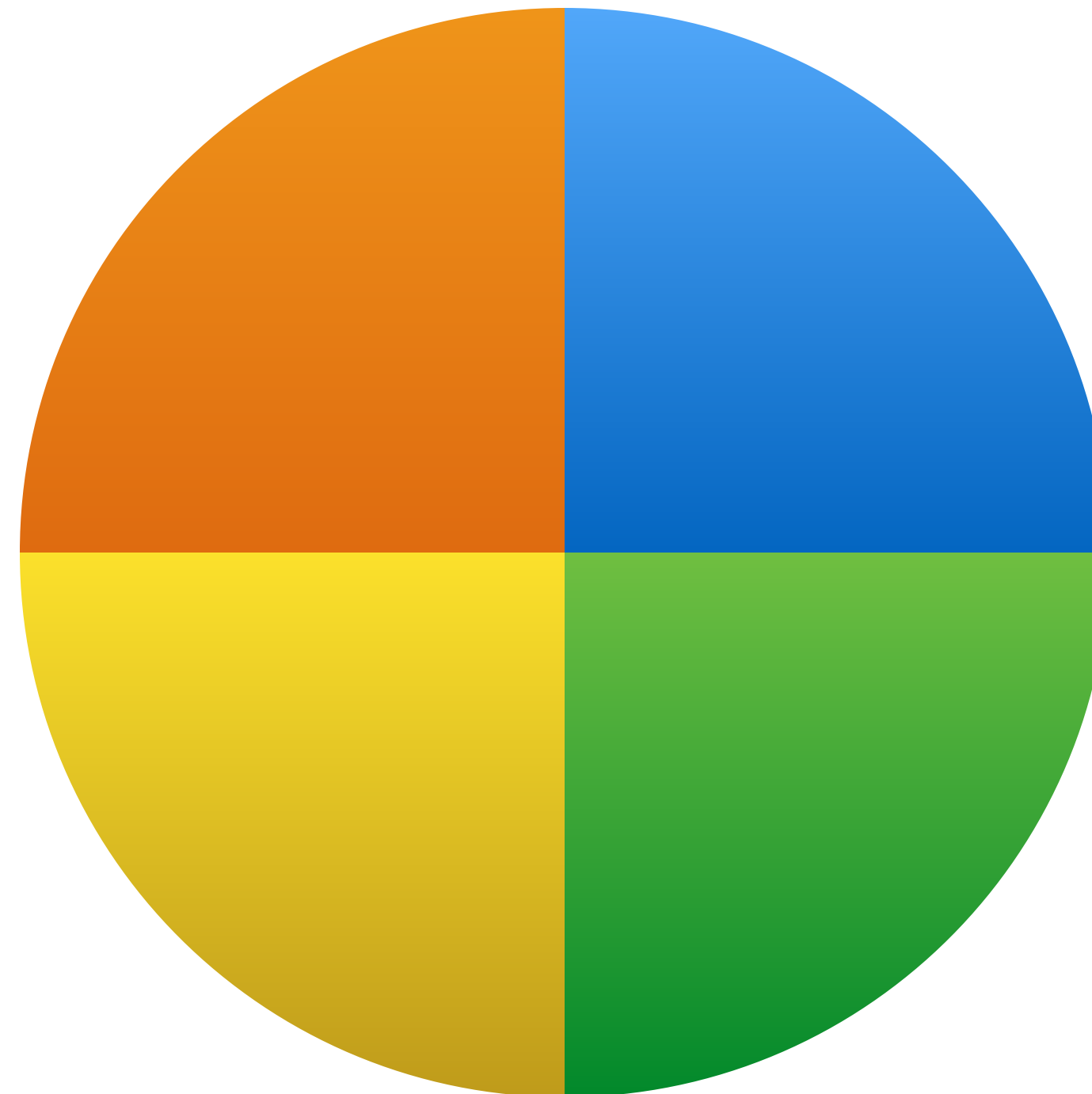
Most Directed

Least Directed

Phases

Procedures

Ideas



Discipline

Focus

Cycles Repeat

relentless reassessment

“It is forbidden to distract or annoy”

–F.I.D.E. Laws of Chess 11.5

Bowling Has A Season

- Software has naturally occurring seasons that usher observable changes in behavior and activity
- Visibility into phases helps others know when to submit requirements, or when to review a nearly completed release
- Cycles provide for relentless reassessment of design and process



New York City Taxi Cab Drivers

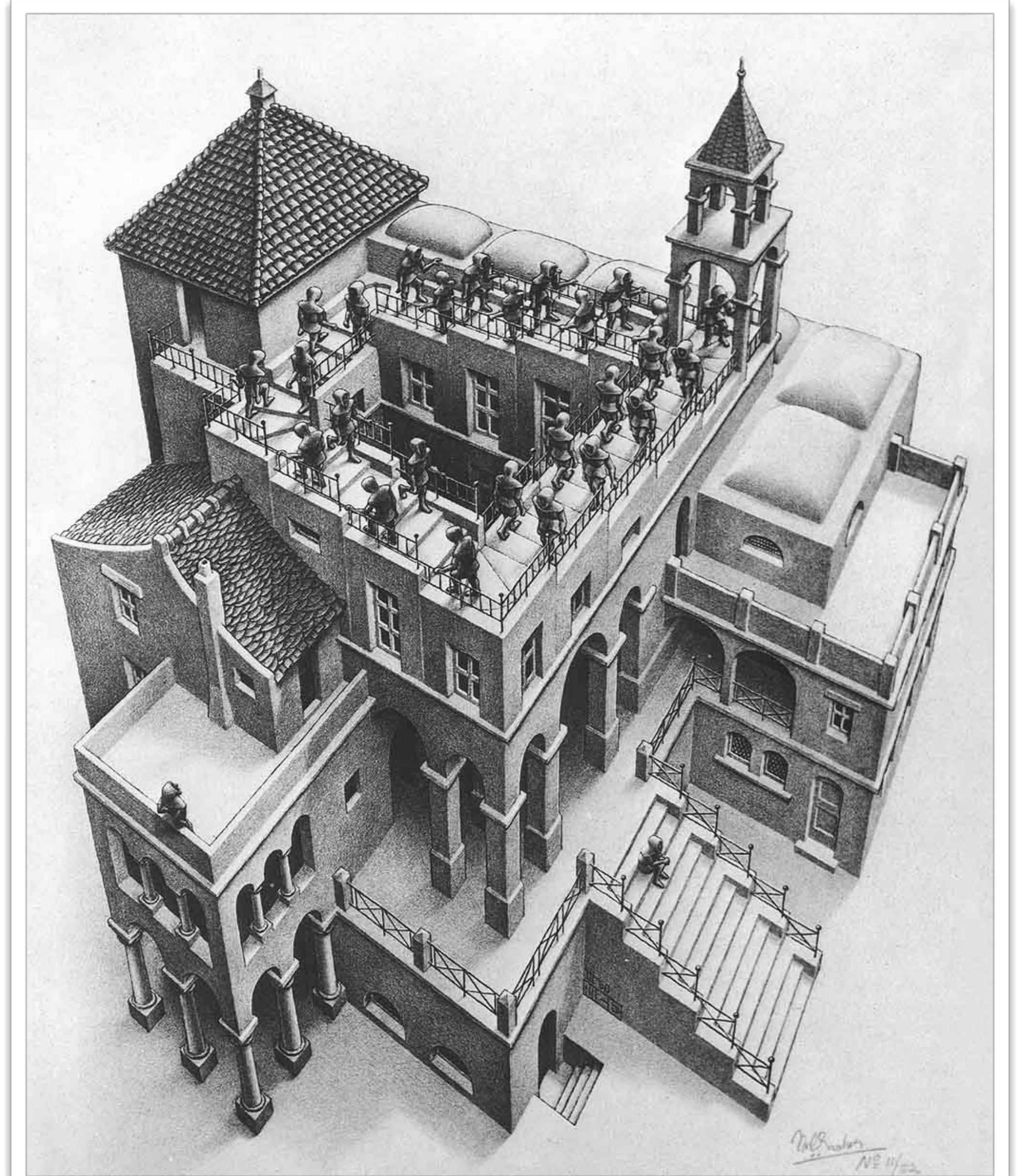
Software Is Unencumbered By The Laws Of Physics

Taxis Have Limits

Imagine two people take a cab ride from the same point in the city...

What If We Could See Software Design?

In the real world, we don't see
people building staircases into
walls, or doors leading to
nowhere



“some measures that are plainly sensible from the real-world perspective, are simply not valid for software construction.”

–Beyond Agile

Compared to real-world professions such as Taxi drivers, the effectiveness of software engineers has a far greater spectrum of performance.

New York City Taxi Cab Drivers

- In the real world, where construction is visible, there are clearly sensible ways of making things
- In software, design and invention are the true levers affecting progress, they can change the entire construction landscape
- Hours worked, even tasks completed are not indicators of underlying software design and thus prove to be unreliable standards of progress

An Iceberg Lays 90% Unseen

This Relationship Has Been Proven By Science



“In software there is no relationship between the visible artifacts and the true underlying complexity of that system...”

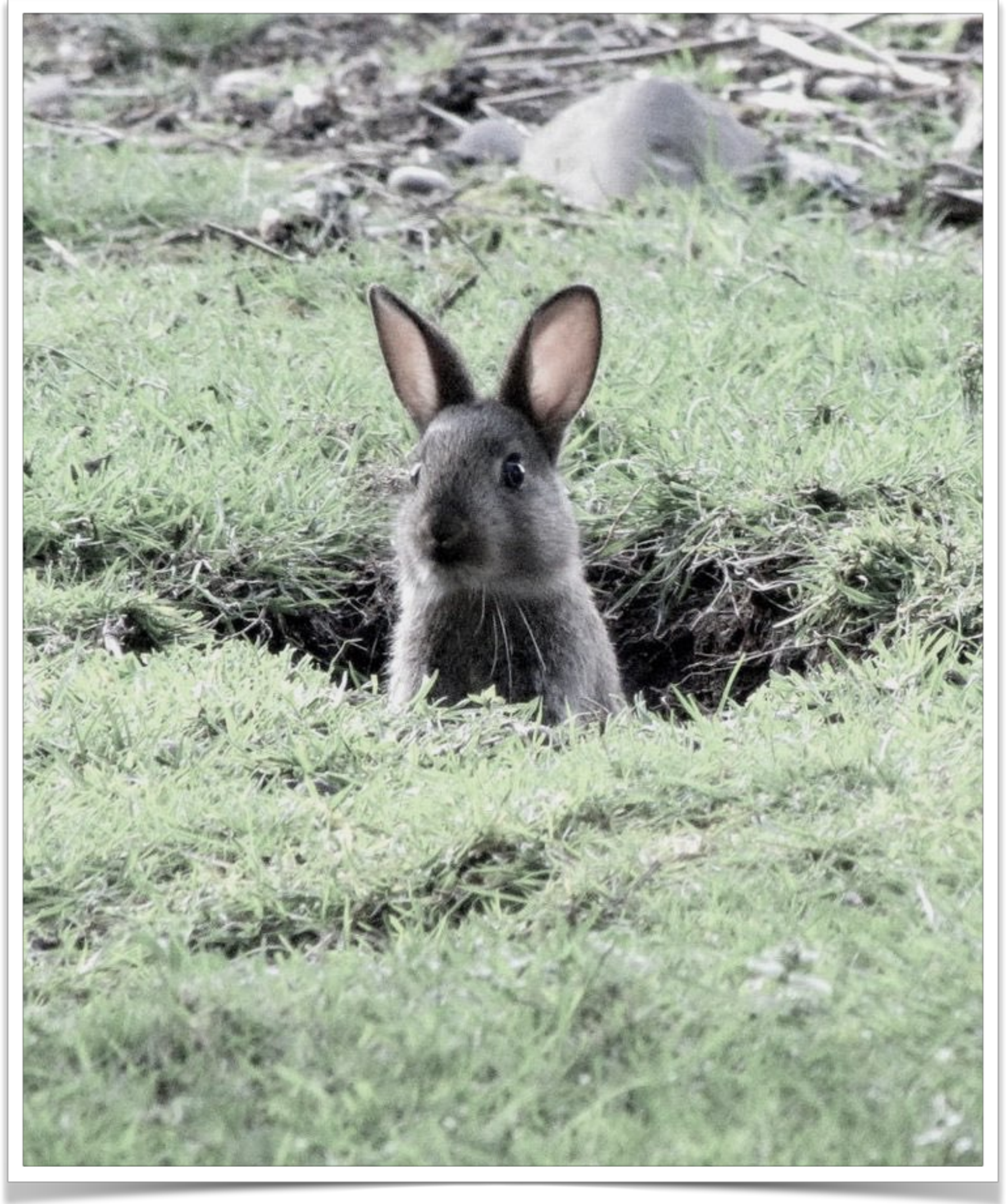
–Beyond Agile

“Software is always in motion, it remains in a balanced equilibrium — building up with new features and functions until it begins to collapse under its own weight. We fight back against this collapse using *design*.”

–Beyond Agile

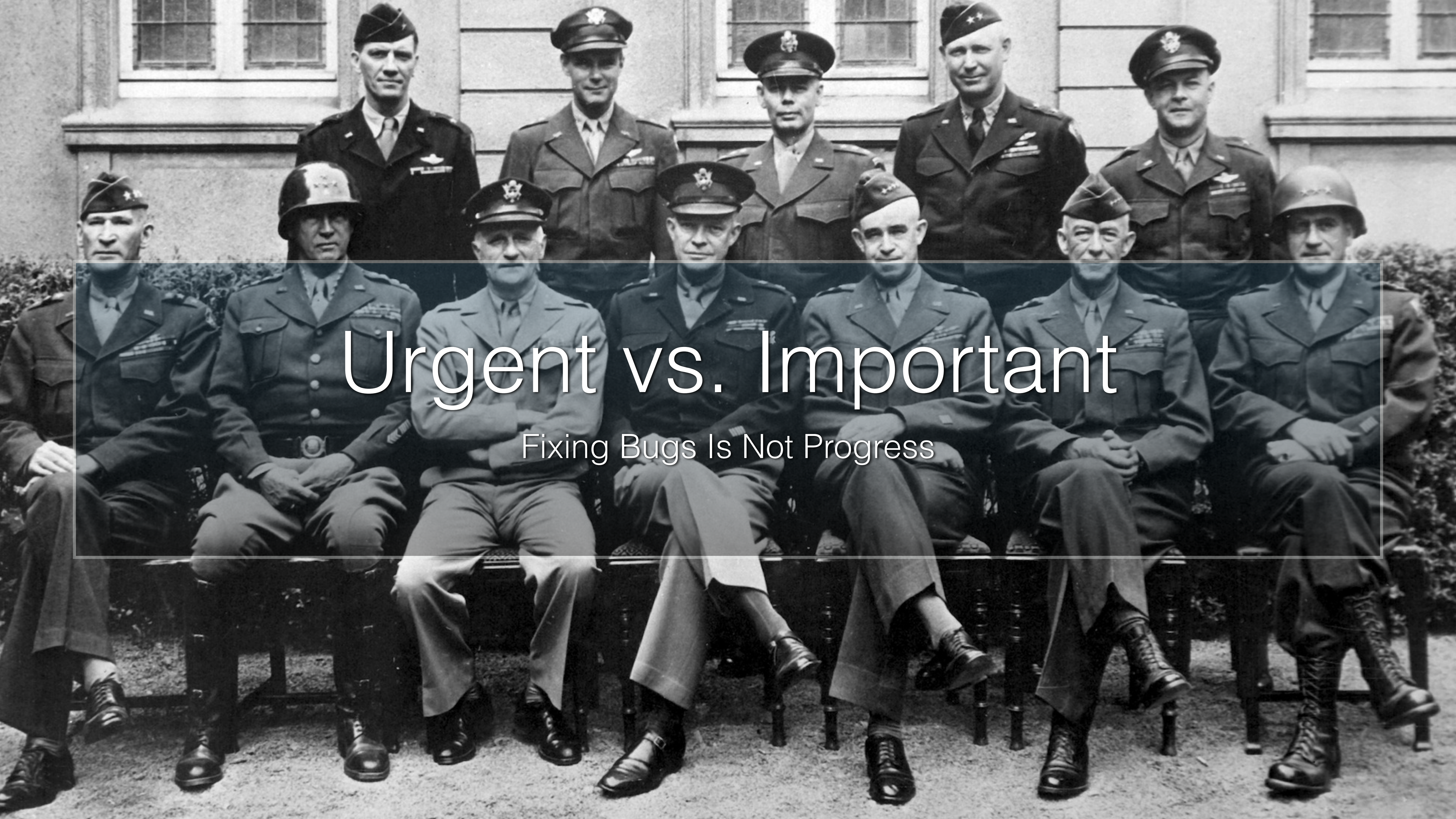
The unseen 'other 90%' goes deep

Project Management tools track
things people can perceive, and
interact with



An Iceberg Lays 90% Unseen

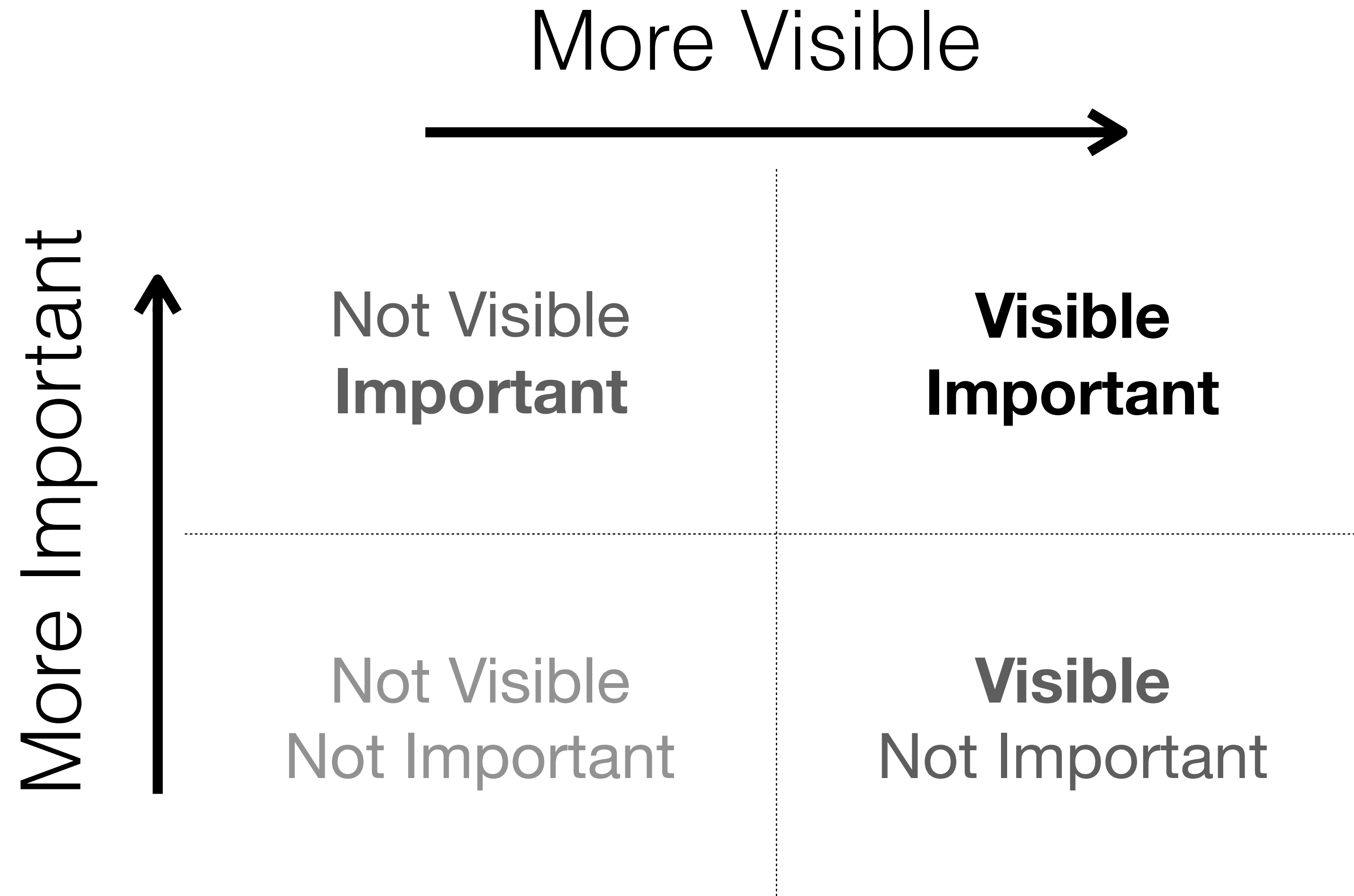
- Project management tools track what gets reported
- Unscientific interpretation and over-reliance can confuse activity with progress
- Design is the true lever effecting software construction and can change the entire construction landscape



Urgent vs. Important

Fixing Bugs Is Not Progress

Urgent vs. Important

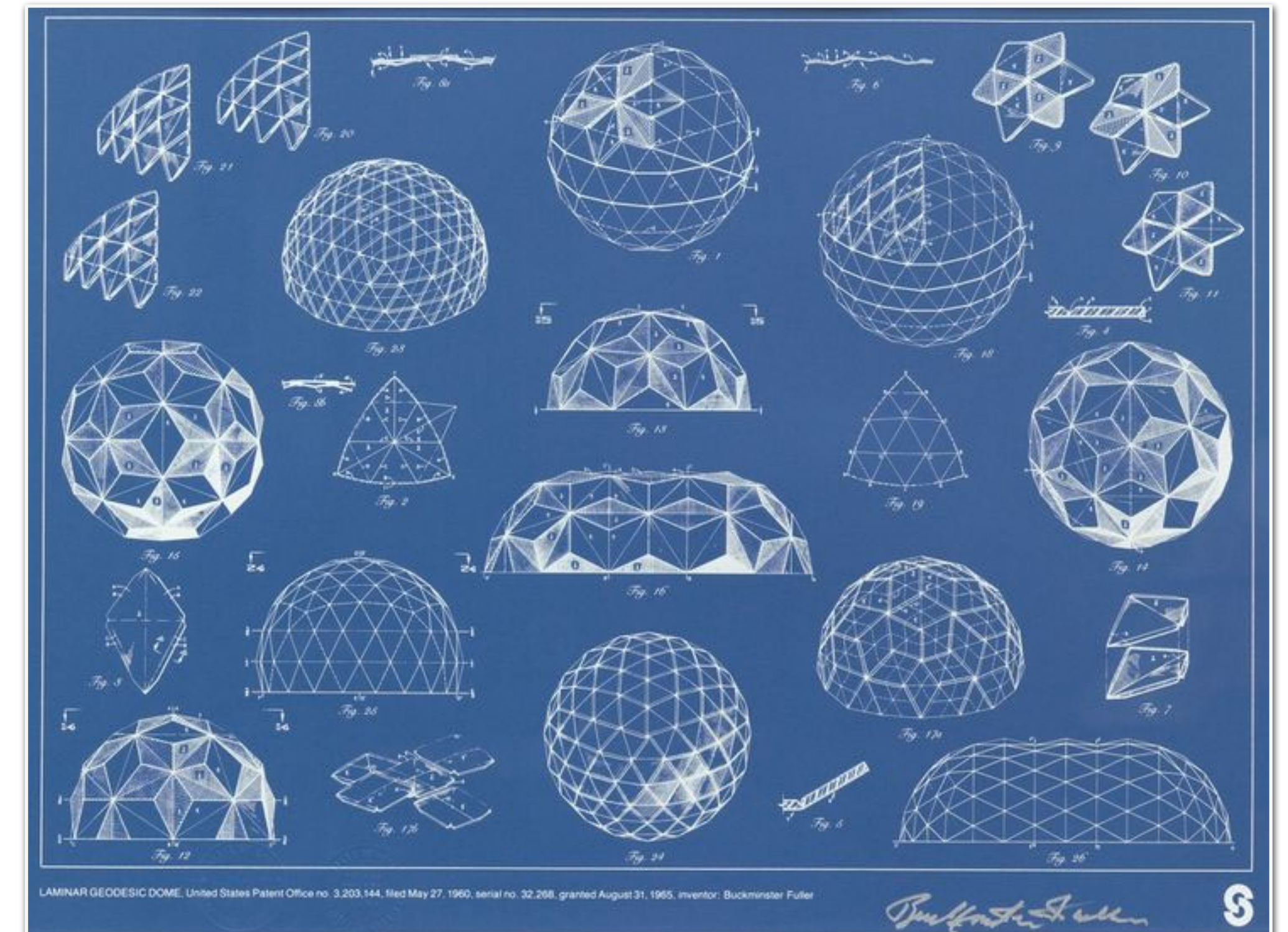


“Program testing can be used to show the presence of bugs, but never to show their absence!”

–Edsger Dijkstra, Dutch Computer Scientist

“Architectural”

Bugs or ***Features*** where
stability of design is critical, and
complexity of function merits
something of a blueprint



Every project has what we might call “***Architectural Features***” or complex endeavors. Every project has what we might call “***Architectural Bug-Makers***” or delinquent code.

“a code smell is a surface indication that usually corresponds to a deeper problem in the system”

–Martin Fowler, British Computer Scientist

Refactoring Is Part of Everyday Life

One cannot replace the retaining-wall of a reservoir and leave it partway complete

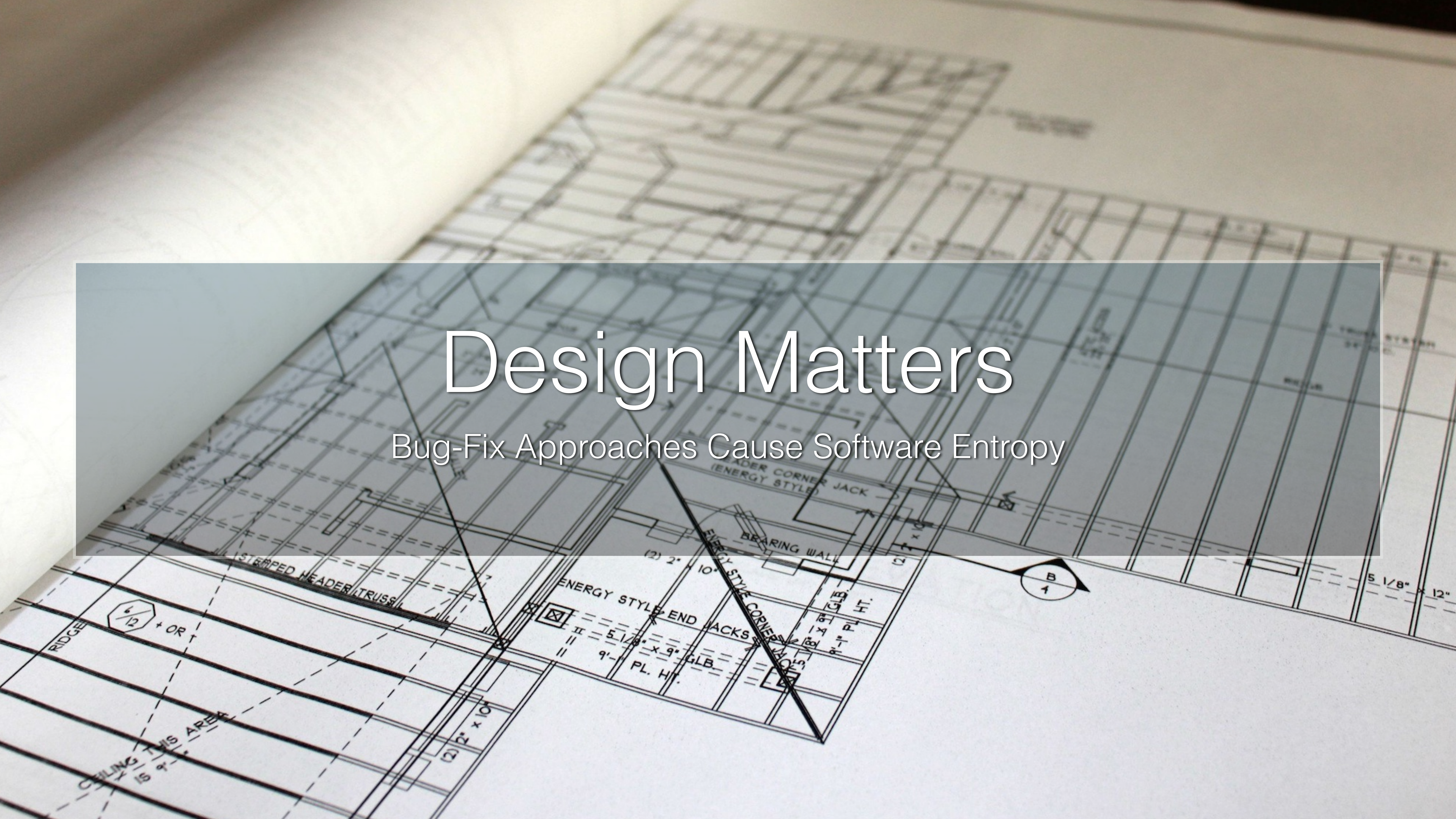


Urgent vs. Important

- In software; *urgent* is what people see and *important* is good design
- Code smell is useful, but the team has already given a snarky nickname to the culprits of code that are delinquent
- It takes science to assess the value of refactoring

Design Matters

Bug-Fix Approaches Cause Software Entropy



“Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves.”

–Alan Kay, American Computer Scientist

Bug-Fix Approaches Cause Software Entropy

Ad-hoc bug fixing results in a patchwork effect that degrades design layers into a more disordered state



Design Matters

- Cathedrals use a million times less stone than a Pyramid —the difference is not material, it is knowledge
- Architecture has even greater meaning in software than it does in the everyday world
- Unordered bug fixing misses an opportunity to assess the value of refactoring faulty design



Balance & Serialize

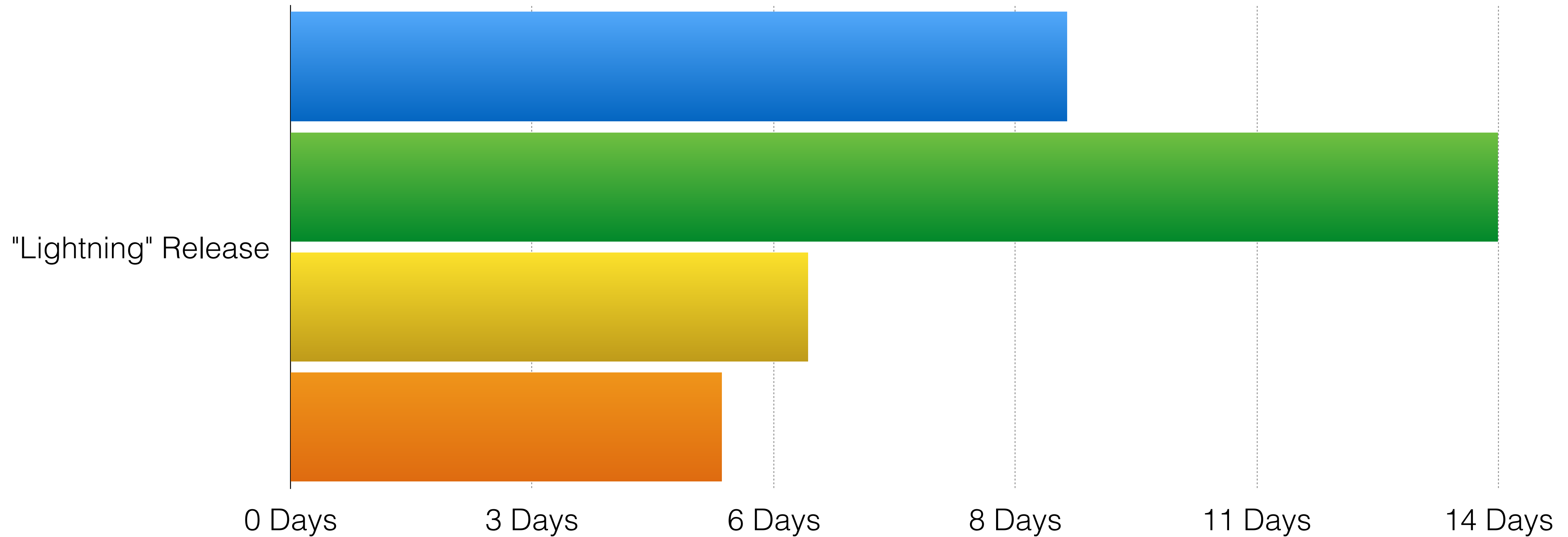
Balance First

“Because Cycles repeat phases in order, we can determine the boundaries of our schedule as surely as predicting spring follows winter.”

–Beyond Agile

Balance

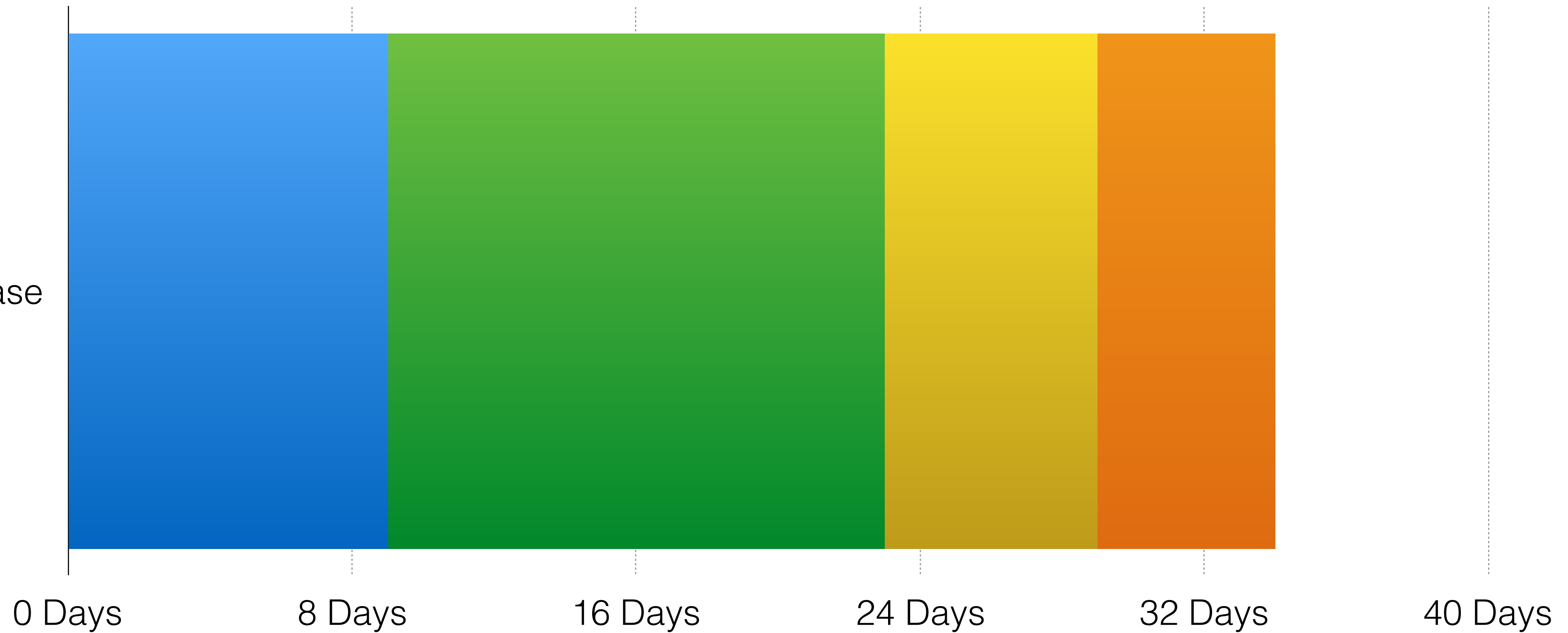
■ Design ■ Develop ■ Debug ■ Deploy



Serialize


■ Design ■ Develop ■ Debug ■ Deploy

"Lightning" Release



Balance & Serialize

- Start with a name, theme and handful of architectural elements, adjust estimates with each team
- Phases repeat in order, we can determine the boundaries of our schedule as surely as predicting spring follows winter
- Natural self-governance is formed by unanimous support of worthy and authentic schedules



Are We On-Time Yet?

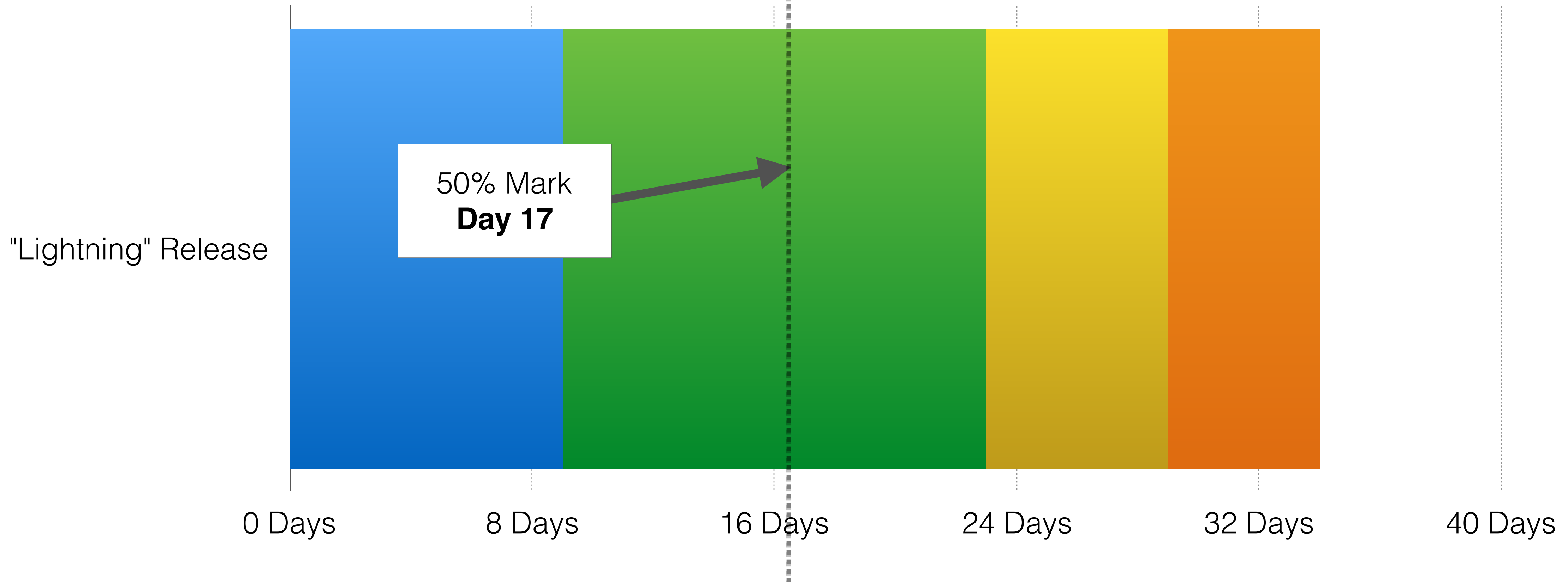
The Current Phase Is Bellwether

“Are we on time? How could anybody say firsthand, if software can’t even be seen?”

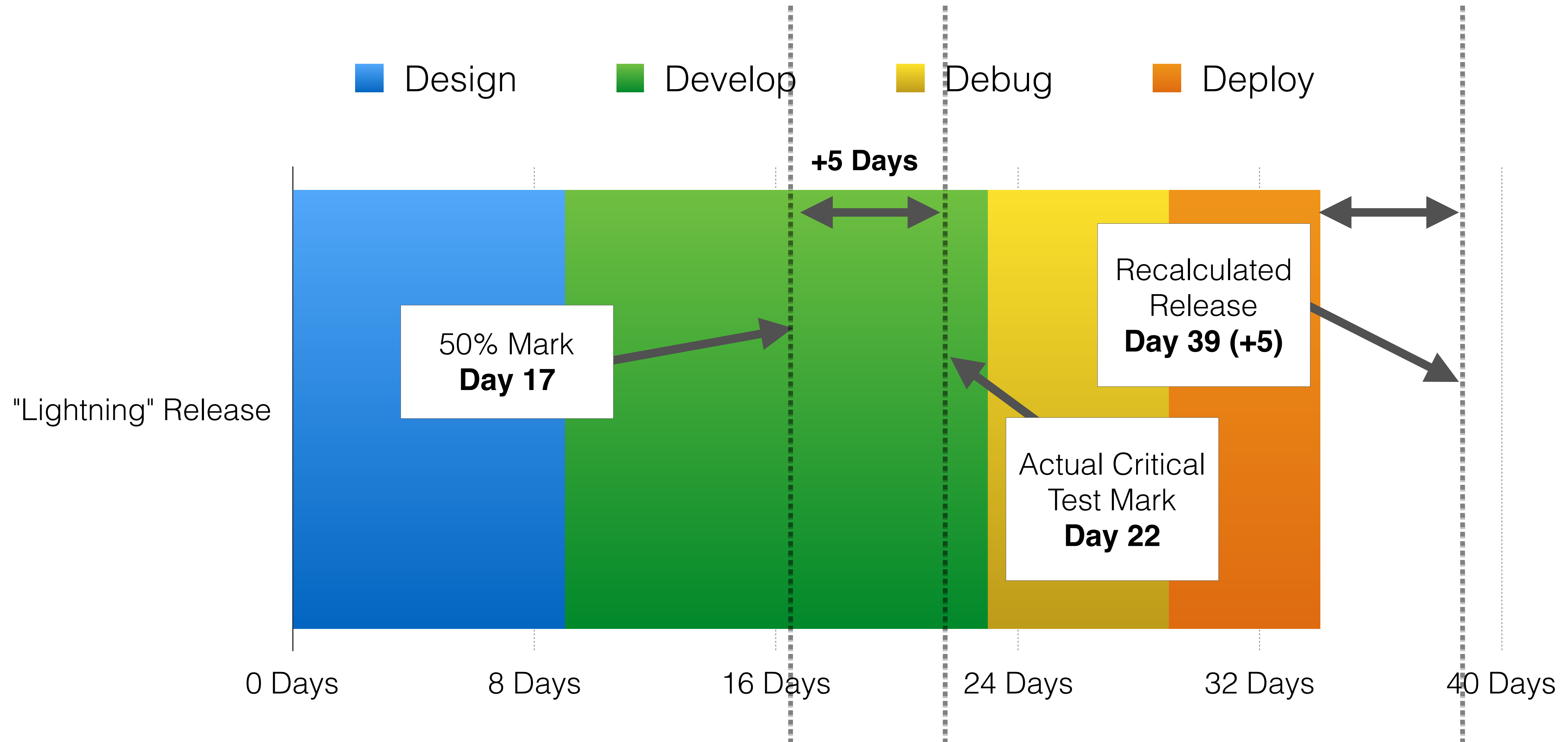
–Beyond Agile

Serialize

■ Design ■ Develop ■ Debug ■ Deploy



Serialize



Sum It Up

1. There are pre-defined milestones set into each Cycle
2. These milestones dictate when estimates can and should be made
3. There are times when we cannot accurately update schedule
4. There are times we must update schedule because milestones were met

Are We on Time Yet?

- Each phase of the software Cycle is identifiable by taking note of the activities in which each team is engaged
- Project Managers can reliably assess *on-time status at a distance*, simply by identifying schedule midpoint and reviewing critical tests
- There are only a few times at which any schedule estimates change



Testing

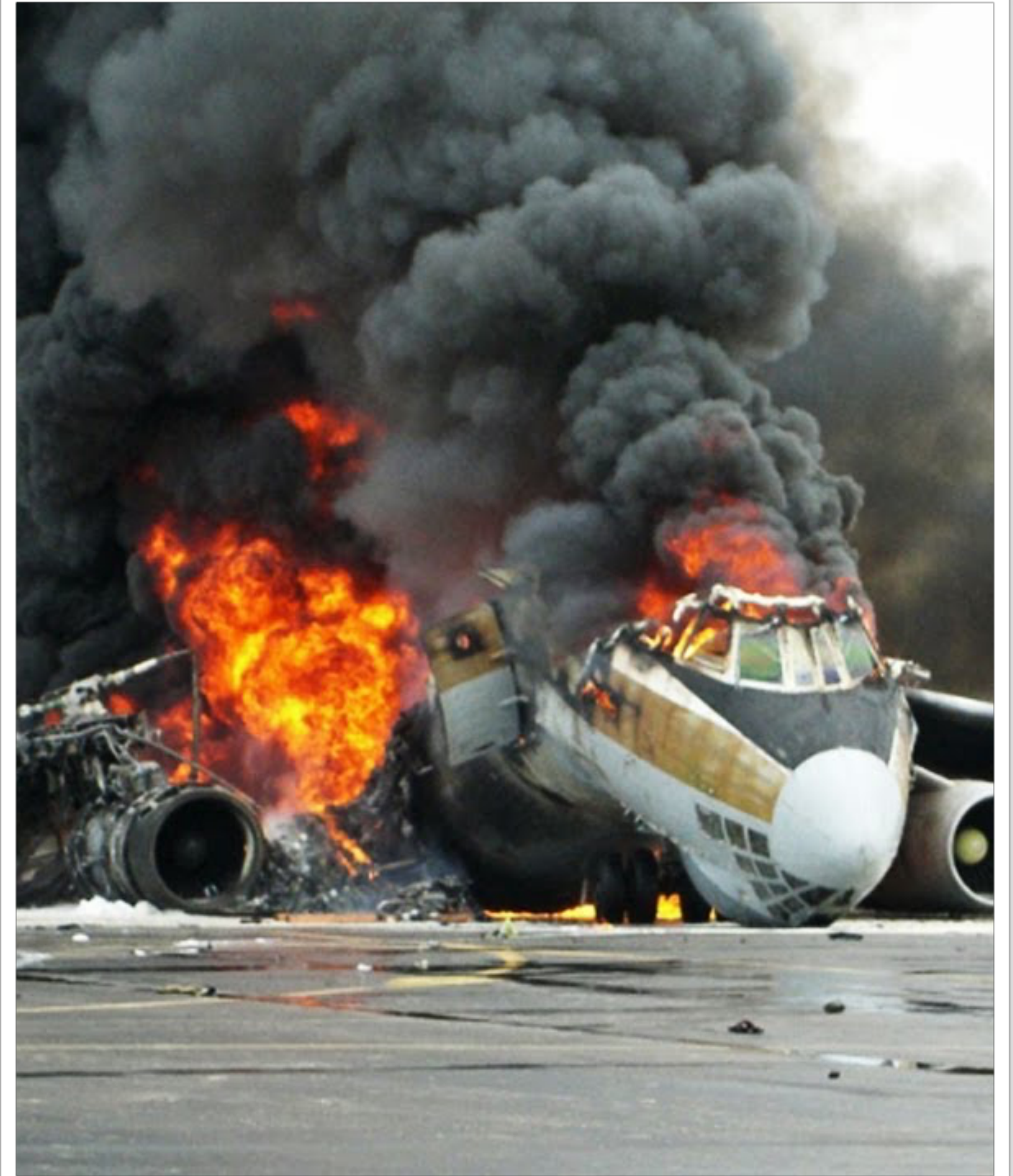
Occupational Safety

“If debugging is the process of removing software bugs, then programming must be the process of putting them in.”

–Edsger Dijkstra, Dutch Computer Scientist

Underwriter's Insurance

quantify and evaluate safety
procedure and make
recommendation based on
science



“If the one-and-only concern of the NTSB were safety, the investigative agency would simply ground all airplanes and close all roads.”

–Beyond Agile

“Proper testing is achieved by appropriately allocating resources to ensure engineers perform their mission in the most cost-effective and friction-free manner possible.”

–Beyond Agile

Occupational Safety

- Testing is occupational safety. It is the National Transportation Safety Board (NTSB) of software construction
- Project Managers are the Underwriter's Insurance, there to economically balance safety procedure using science
- Testing allows software architects to confront complexity with a clarity and composure that comes only from a sense of safety and security



Anatomy of a Release Plan

The Seven Rudimentary Elements Of Communication

7 Rudimentary Elements

1. A central authority to govern approach, a **style guide**
2. Each Cycle has a **release plan**, give it a memorable name
3. **Balance and serialize**—a strategy of schedule ownership and visibility
4. **Release notes** discuss what changed from the original plan
5. Candidate **release docket** lists when and why release candidates were rejected
6. **Post-mortem** outlines what went well, and what can be improved
7. A diary of dated log entries or **lineage-report** contains historic events of merit, spanning the lifecycle of the product

A photograph of the California State Capitol building in Sacramento, featuring its iconic dome and grand staircase. A semi-transparent rectangular box is overlaid on the upper half of the image, containing the title text. The background shows a clear blue sky and some greenery on the right side.

“Software Capital” & The Pursuit of Quality

Forces In Motion

“If we want to be serious about quality, it is time to get tired of finding bugs and start preventing their happening in the first place.”

–Alan Page, American Computer Scientist

What exactly is Software Quality?

Quality is the *will* behind
extending ourselves and our
effort



The Pursuit of Quality

- Quality is the *will* behind extending ourselves and our effort — without it our software construction engine will run dry
- This pursuit of Quality is the flow of energy we tap into for effective turn of phase
- Quality is everywhere, it is the mortar holding the bricks of our Cathedral, the fabric that binds it all together

A close-up photograph of a front-loading washing machine with its door open. Inside, several light-colored towels are visible. The machine has a metallic finish. A semi-transparent dark rectangle is overlaid in the center, containing the text.

What Goes Around

Comes Around

“When we think of cyclic software construction, we don’t envision a cascade, or sprint to a goalpost —but rather an opportunity for tapping into a continuous flow of natural energy”

–Beyond Agile

Forces in Motion

Schedule, urgency and ***quality***
mix together and combine,
providing the fuel to advance our
position, and the control we use
to navigate



***“The call of urgency and the pressure of schedule are held in place
by the power of quality.”***

–Beyond Agile

“Quality. Up yours.”

–Anonymous

Thank You
beyondagilethebook.com

