Pragmatics of Agile Development

Venkat Subramaniam

venkats@agiledeveloper.com
twitter: @venkats
http://www.AgileDeveloper.com

What’s Agile?

- You’re Agile if you have Standup meetings?
- You’re Agile if you write Unit Test?
- You’re Agile if you don’t do any documentation?
- ...

- What makes you Agile?
- More important, Why should you be Agile?

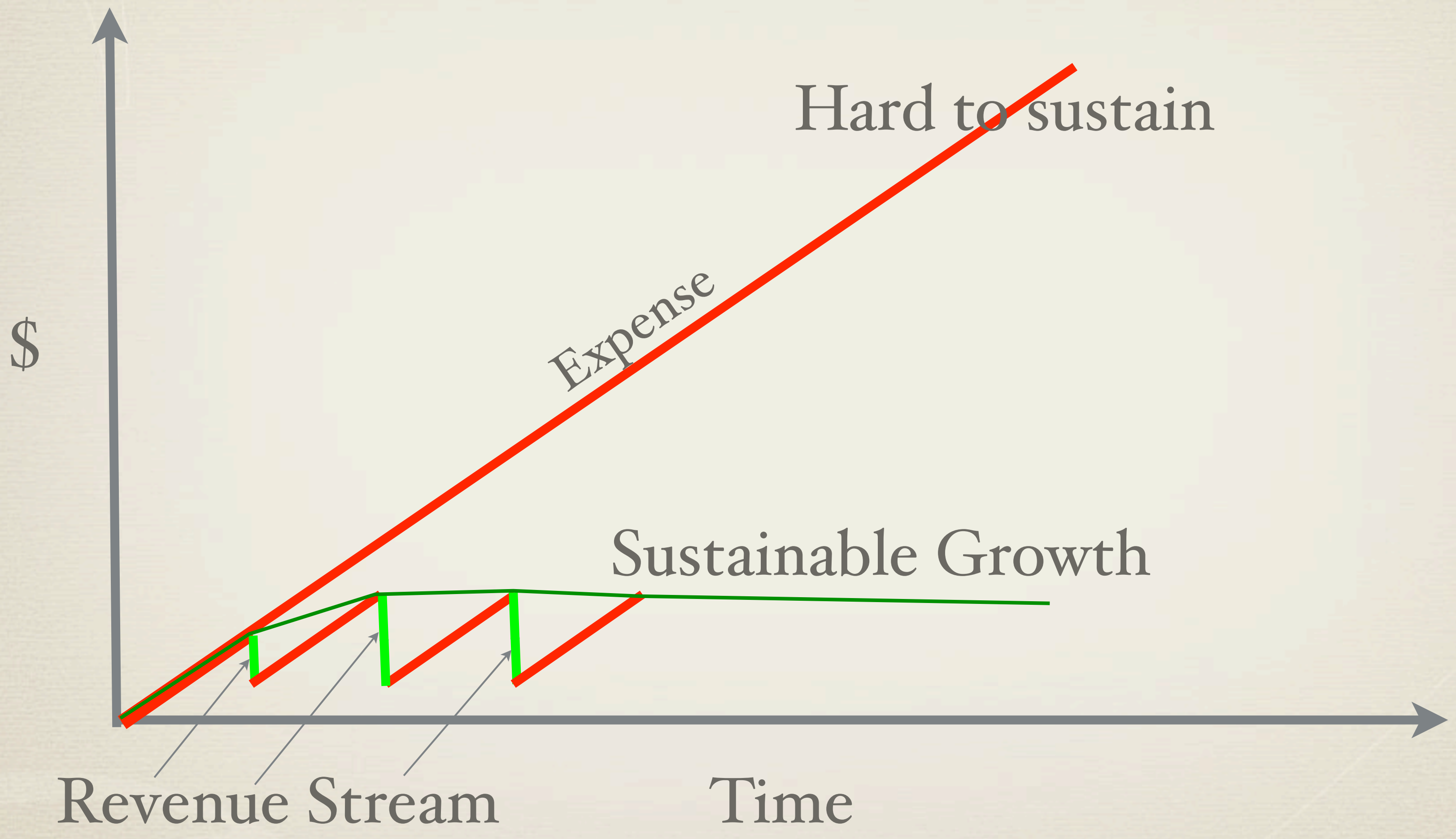
Why be Agile?

- It is about our ability, as individuals, teams, and organizations, to respond to ever changing business conditions
- Change is the only constant
- How can you respond to change?

But Why?

* “It is not the strongest of the species that will survive, or the most intelligent. It is the one most adaptable to change.” – Charles Darwin

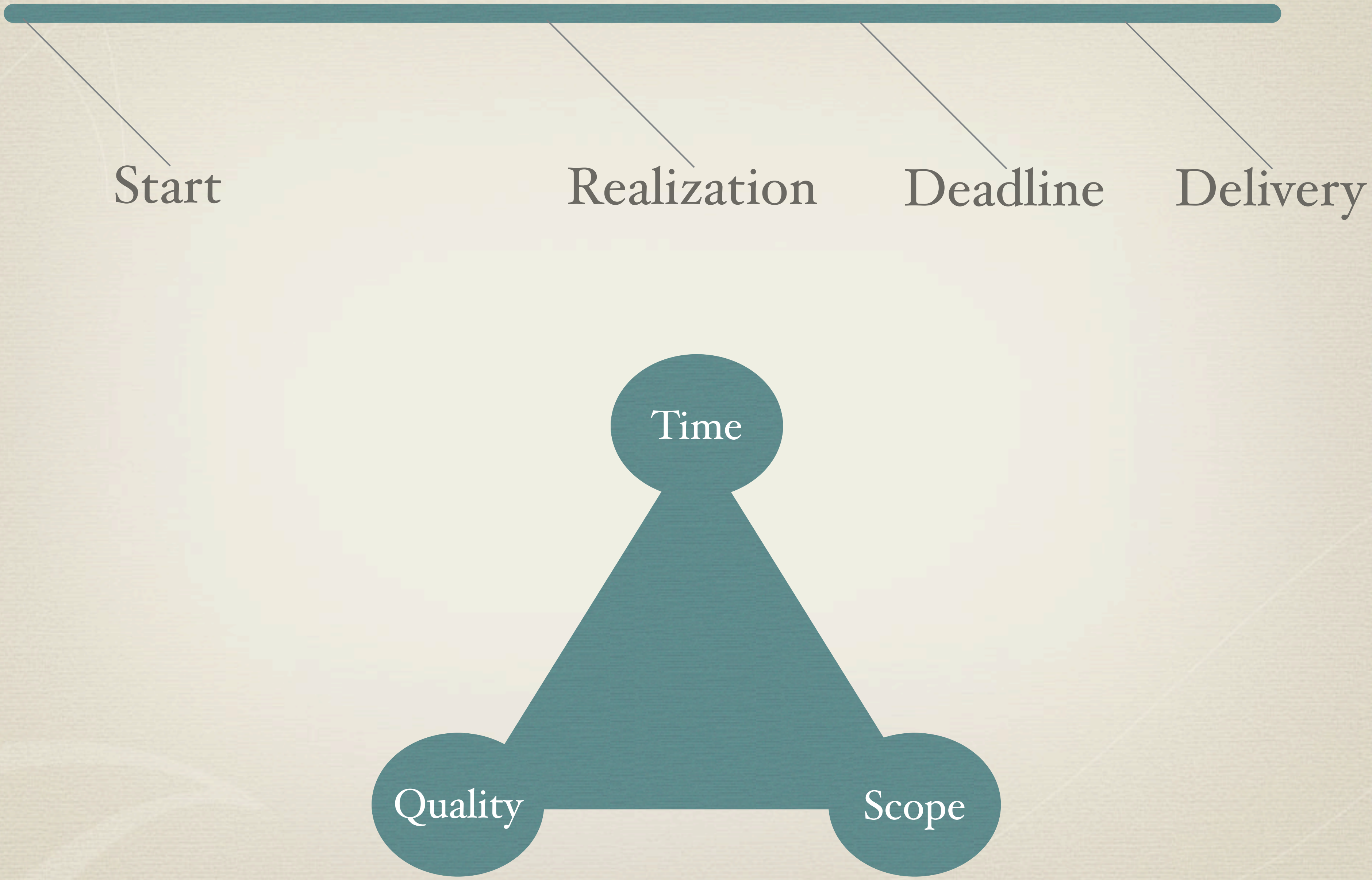
Being Incremental



Can’t Put It on Autopilot

- You can’t put an organization and projects on Autopilot
- The longer you forecast, the larger is your margin of error

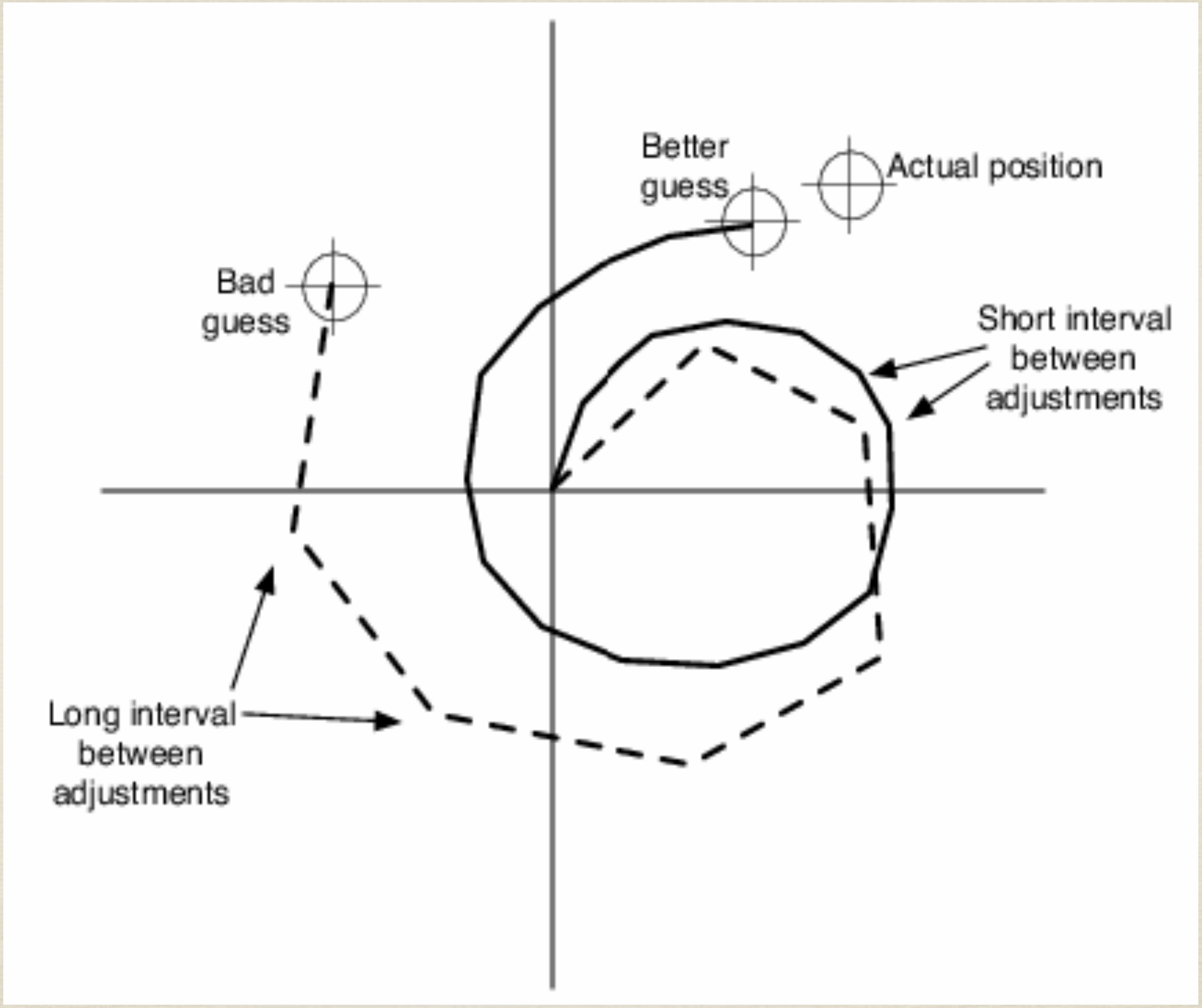
Project & Schedule



Adaptive Planning

- “No plan survives contact with the enemy” - Helmuth von Moltke
- It is more important to succeed than stick with a predefined plan
- Your organization/team/you can dictate only two out of three: Quality, Time, Scope
- What if they/you insist on fixing all the three?
  - Result is failure

Meeting Requirements


Better guess
Actual position
Bad guess
Short interval between adjustments
Long interval between adjustments

Feedback is Critical

- You build some
- Take it to your customers
- See how they use it and what they care the most about
- Make revenue and continue development

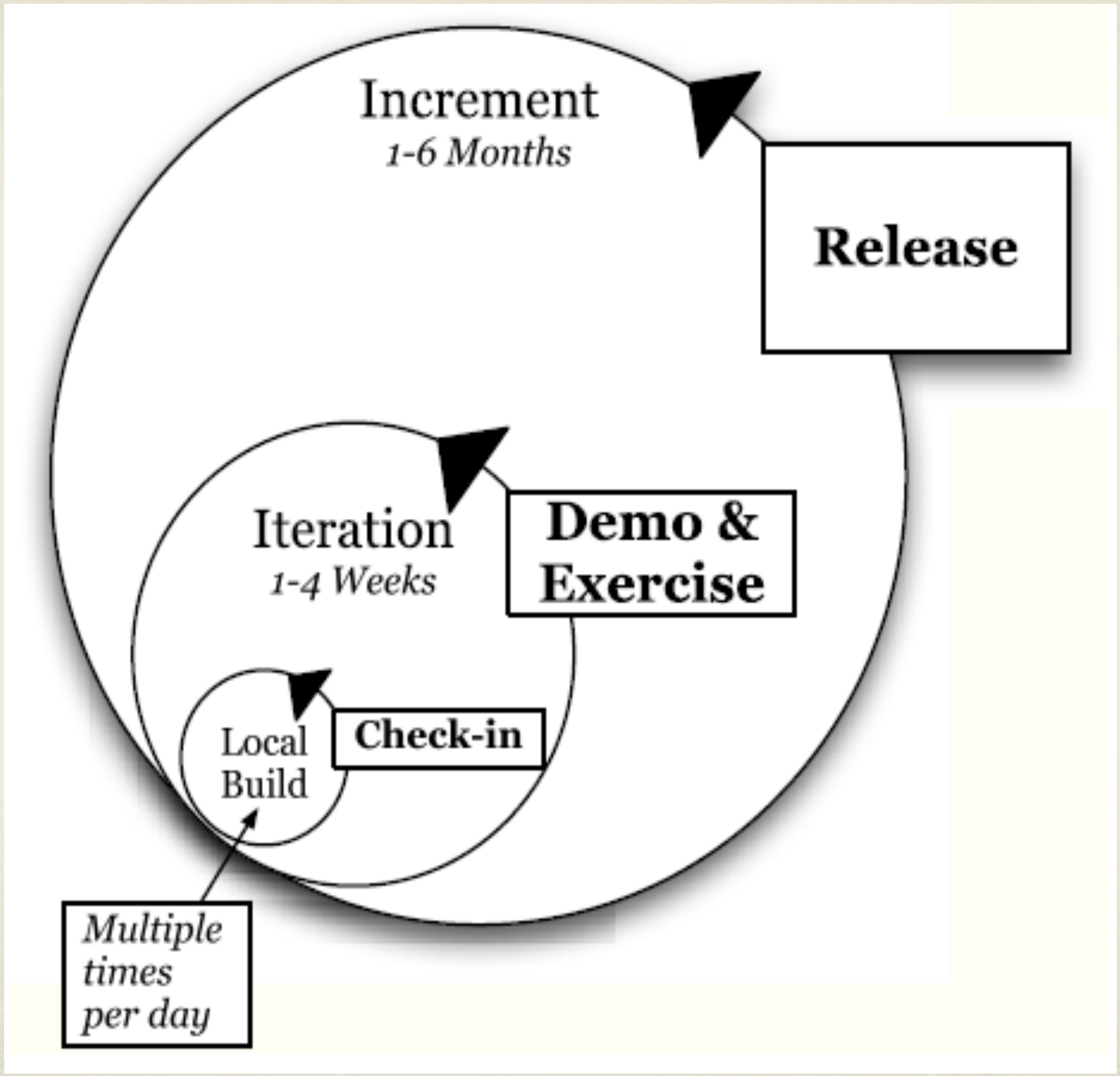
- The requirements you start with may not be the ones you end up with in the final implementation

What you Build?

Software Exhibits Heisenberg Effect

- If your objective is to build what your customers wanted, you will fail
- You need to build what they still want!

Find the Rhythm



What’s Agile?

- It is not about Speed
  - If you focus on Short-term speed, you’d compromise quality
  - That’s sure to bring down your speed relatively soon
  - If you don’t take time to design, to write tests, to get feedback, to make the change affordable, ... that’s like ignoring daily hygiene to make quick progress

What does it take?

- Lots of Discipline
- Hard work
- Adaptive planning
- Retrospection, reevaluation, realization, readjustments, ...

Why is this so hard?

- Software Development is a human activity
- Humans are very creative, but...

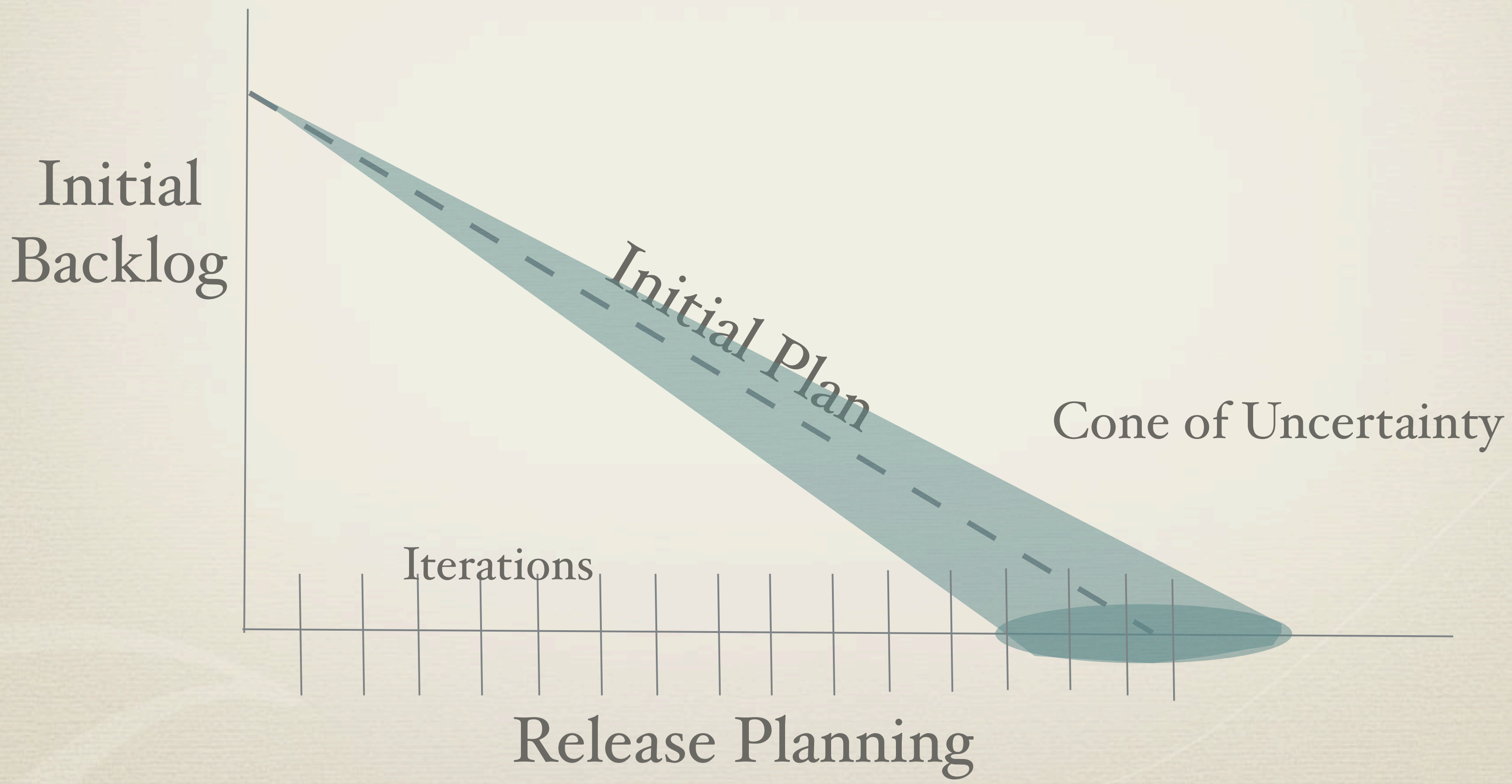
- Change is hard
- Emotional
- Influenced by past experience
- Discipline is hard
- Influenced by pressure, expectations, ...

What about other fields?

- In every human activity, in every field, we took time to learn
- We made mistakes
- We got it wrong
- Then we spend time brooding over it
- Eventually we figure out what works
- From time to time we still make mistakes
- Software Development is such a nascent field... we still have long ways to go

Planning

- Agility is not about lack of planning
- It is adaptive planning



Adaptive Planning

- The planning happens constantly, during each iteration
- Your management team needs to constantly understand the realities and adjust scope or time
- If they expect everything to go as originally planned, the name for that is not agile. It called being in denial
- Engage your management and customers

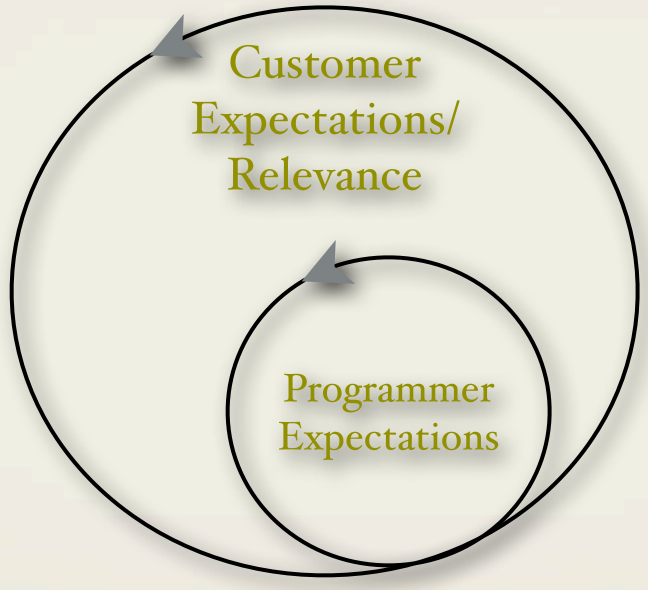
Ask what's Right?

Apply good principles, review constantly, test rigorously

- Are you building the software right?
- Are you building the right software?

Actively seek feedback, ask your application to be exercised, integrate continuously, **run automated functional tests,** take smaller bites

Feedback Driven



Which is More Important?

- The outer circle tells that your code is meeting your customers expectations—obviously that is the most important, right?
- Yes
- So, what if we only focus on that—Let's show it to them often (demo) and ask them to use it (exercise)

Then What?

- Your customers really begin to get the idea when they see the application you've built
- Now they tell you what they really want, what they really care about,...
- You ask your team to change the application accordingly, and then,...

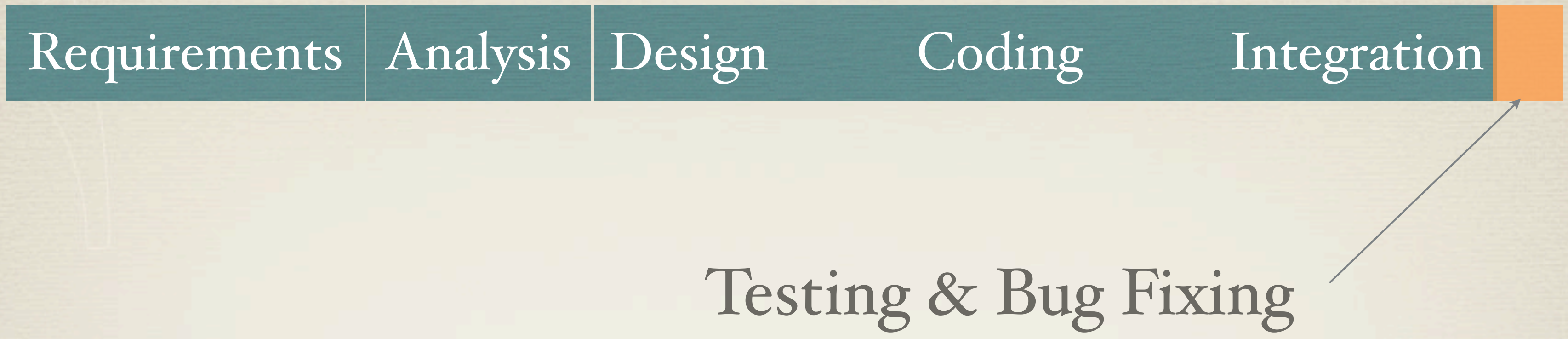
This Leads to Whac-A-Mole Systems

- Your team fixes the part based on the feedback
- Your customers try it out, only to find another unrelated part is found broken
- Your team fixes that and customers now find some other part is broken

Again, Which is Important?

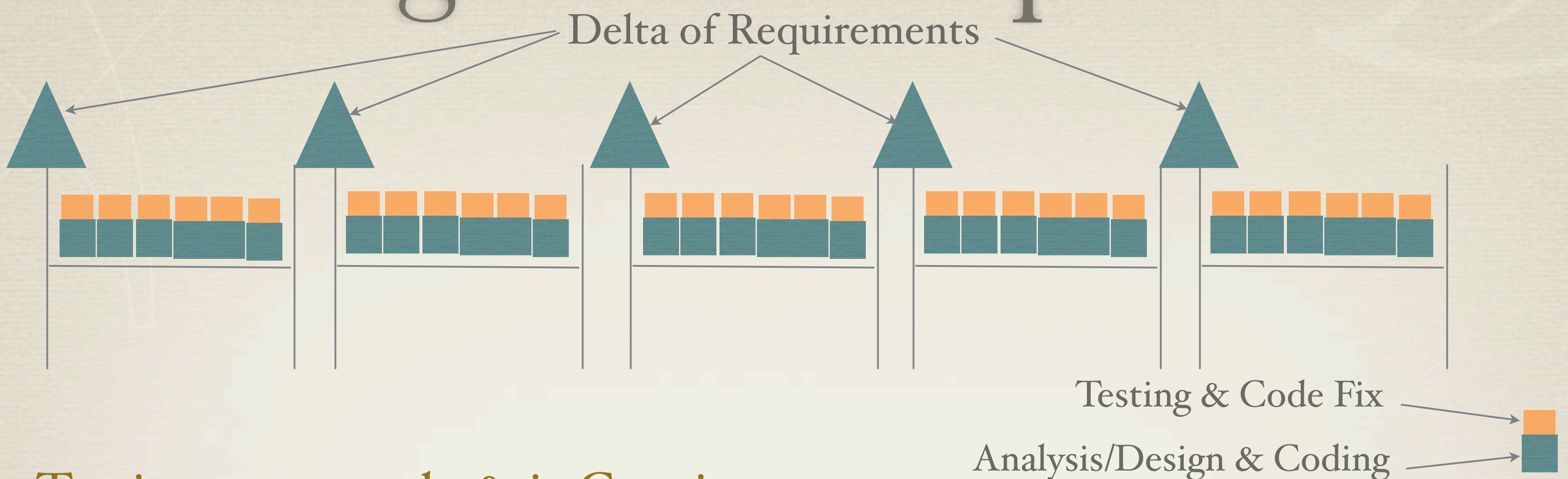
- The inner circle of behavior is required for the outer circle of relevance to be sustainable

Traditional Testing



- Too late in the game
- Often pressure to release
- QA become defenders
- Often looked at as adversaries

Agile Development



- Testing starts early & is Continuous
  - Don't wait until end of iteration to test–test frequently and regularly
- Application is exercised constantly, no surprises later
- QA become support
- Not adversaries, become part of the team
  - Work with customer and programmers—co-located with them

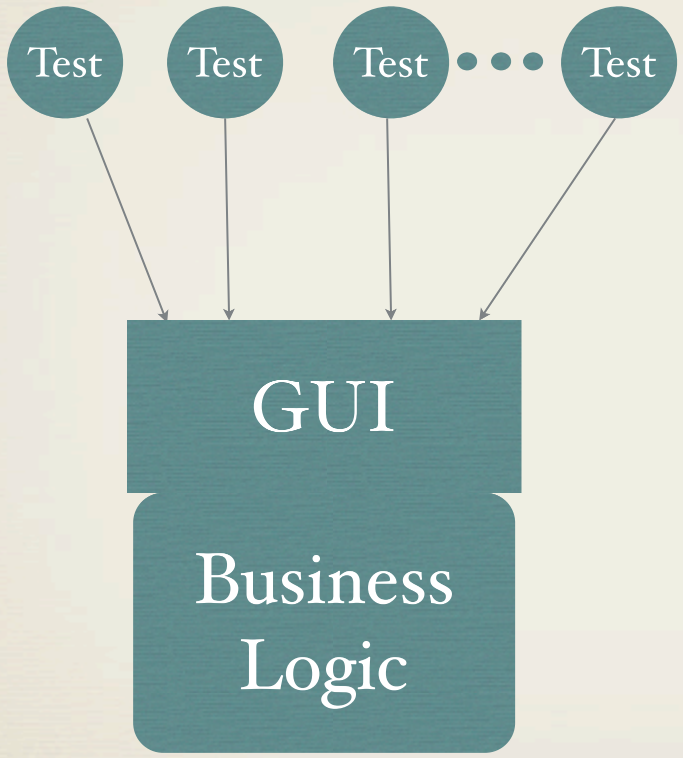
Tenet Of Testing

* As a tester, your responsibility is to author tests, not to run them!

Why Automate Tests?

- “Error rate in manual testing is comparable to the bug rate in the code being tested.”—Boriz Beizer.

Where not to Automate!

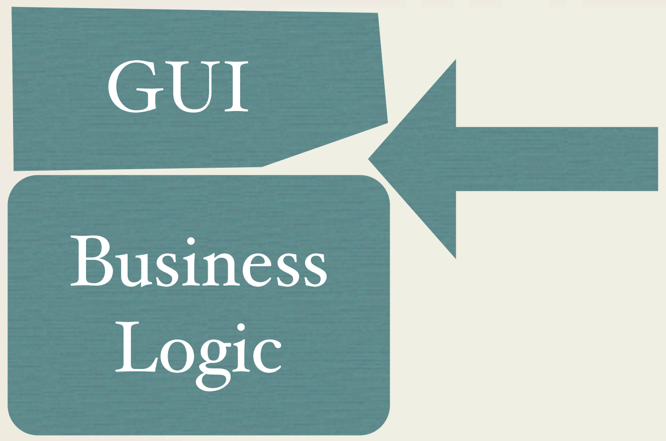


Very hard to automate

Not Effective

Too Brittle

Where to Automate



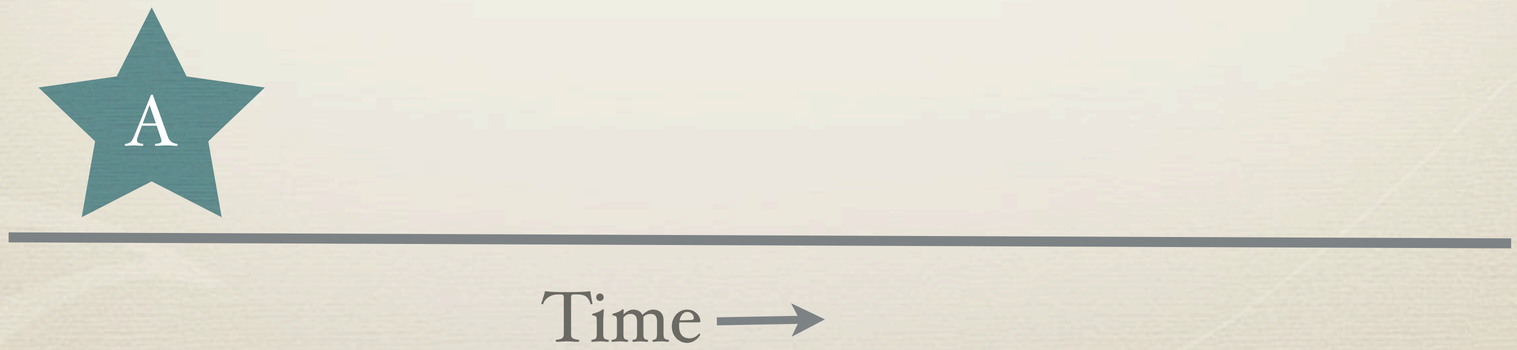
Run Automated Tests
one level below GUI

Documentation

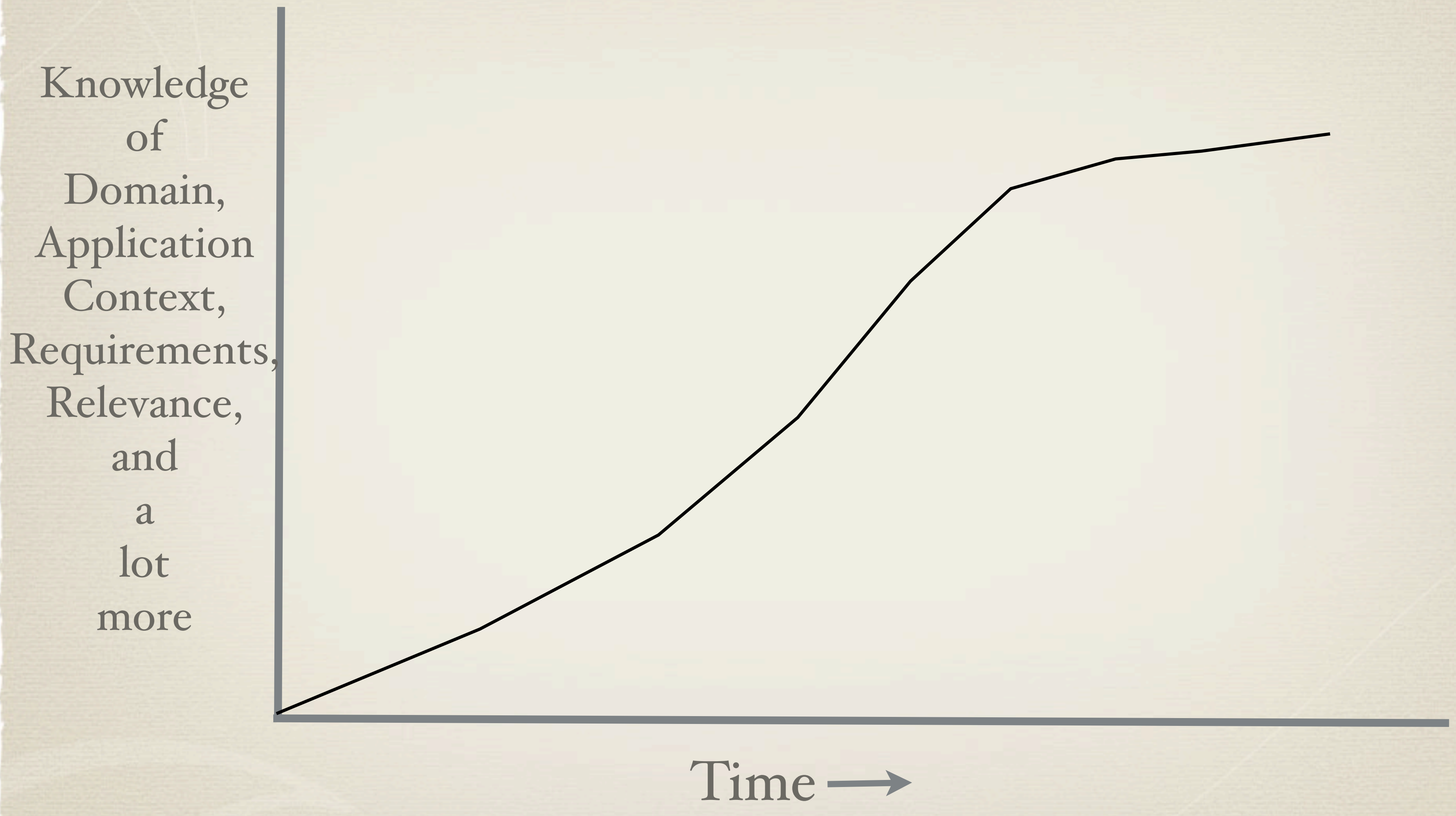
- Is there a place of Documentation in Agile Development?
- Yes, but ...
- Make sure that documents you create are useful *and* really used
- A high level *short* architectural document is necessary
- Unit Tests document tactical design
- Functional Tests are Executable Documents

Architecture

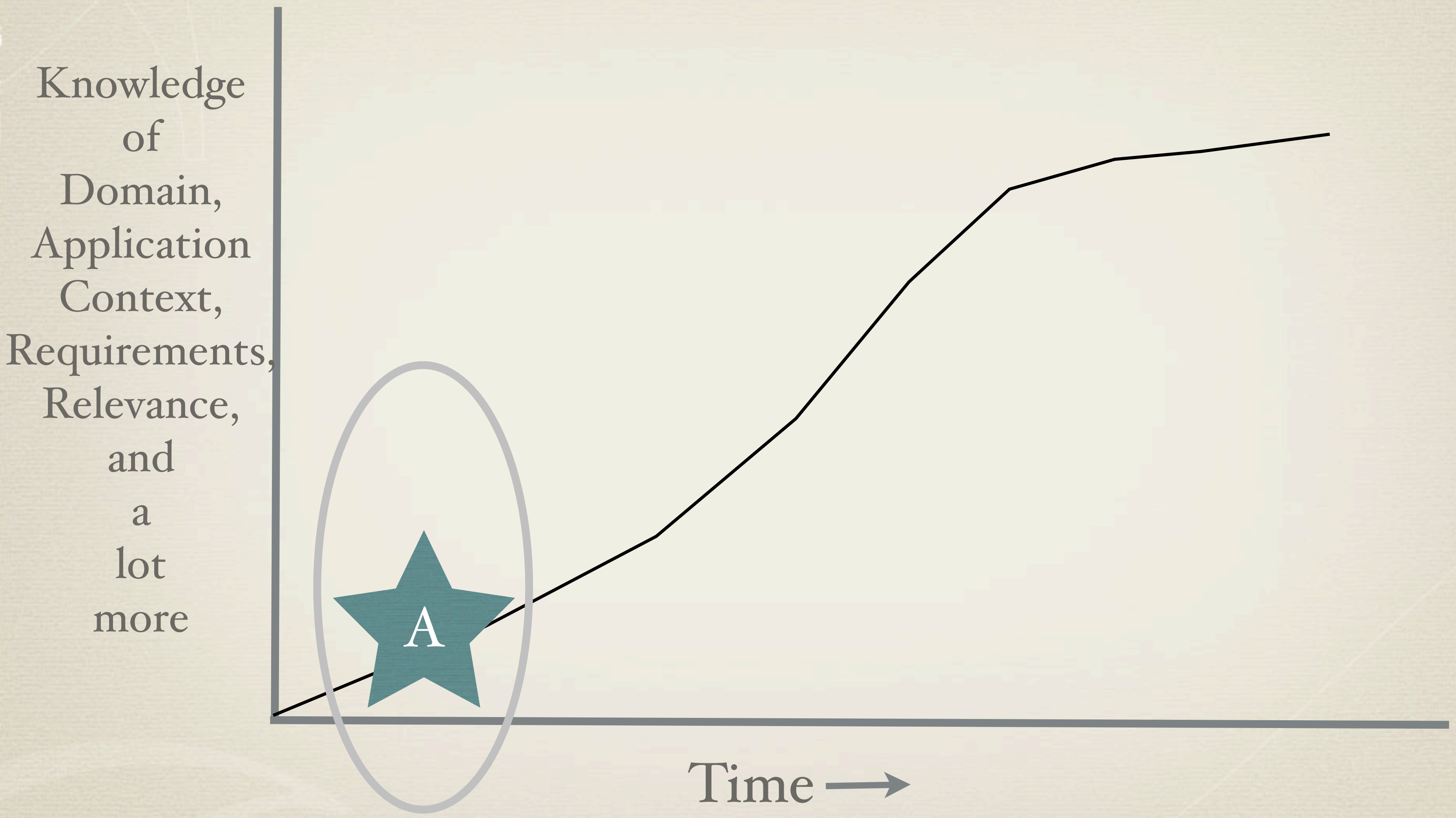
- Very significant
- Need to get it right
- When do you typically develop Architecture?



What we Know?


Knowledge of Domain, Application Context, Requirements, Relevance, and a lot more
Time ⟶

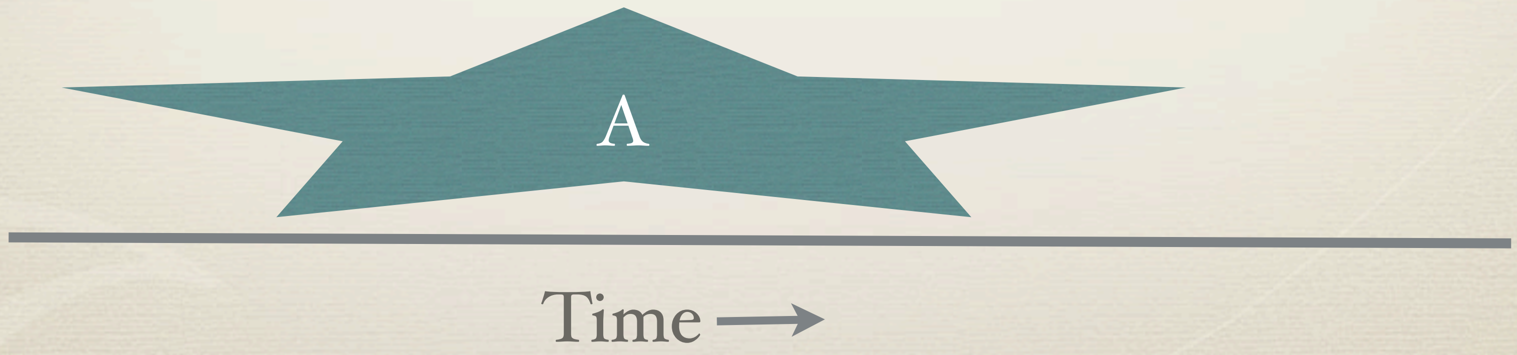
Visit that Again



One word that describes this: RISK

Why Evolutionary Design?

- Why would you take on something that important when you know the least?
- You don't want to get it wrong—so don't get it when you don't have a clue



Unit Testing

- Programmer Activity
- Not really about testing to see if code works
- It is about documenting so your code continues to work as code changes and system evolves
- But, my boss wants speed, can I just code?
- If you ignore, don't expect that speed to continue

Cost of Unit Testing

- You're going to write about 2 to 3 times unit test as code under test
- It takes time, effort, money
- It takes a lot of discipline
- It is a skill you've to develop
- It is not an insurance, it is an investment–it pays off in big dividends

Refer to study by Dr. Laurie Williams

Who's doing it?

- Slowly gaining acceptance
- It is like exercising
- Most people will accept exercising is good for health, but only a few do it
- Unit Testing is software equivalent of exercising

Iteration and Demo

- Most Agile books tell us "Demo at the end of Iteration"
- Let's think about that for a minute
- If you show what you've done to your customers only at the end of each iteration, what are the chances your iteration will succeed?
- Yes demo at the end of iteration–that's a nice ceremony
- But, consult with them constantly

Iteration and Demo

- As you develop, demo and consult with customers
- You can mock things to get feedback
- Show partial solutions
- Do what it takes to get their input and feedback

- Use end of iteration demo as final feedback and closure
- Do not build for this demo. Demo what you've built at this time.

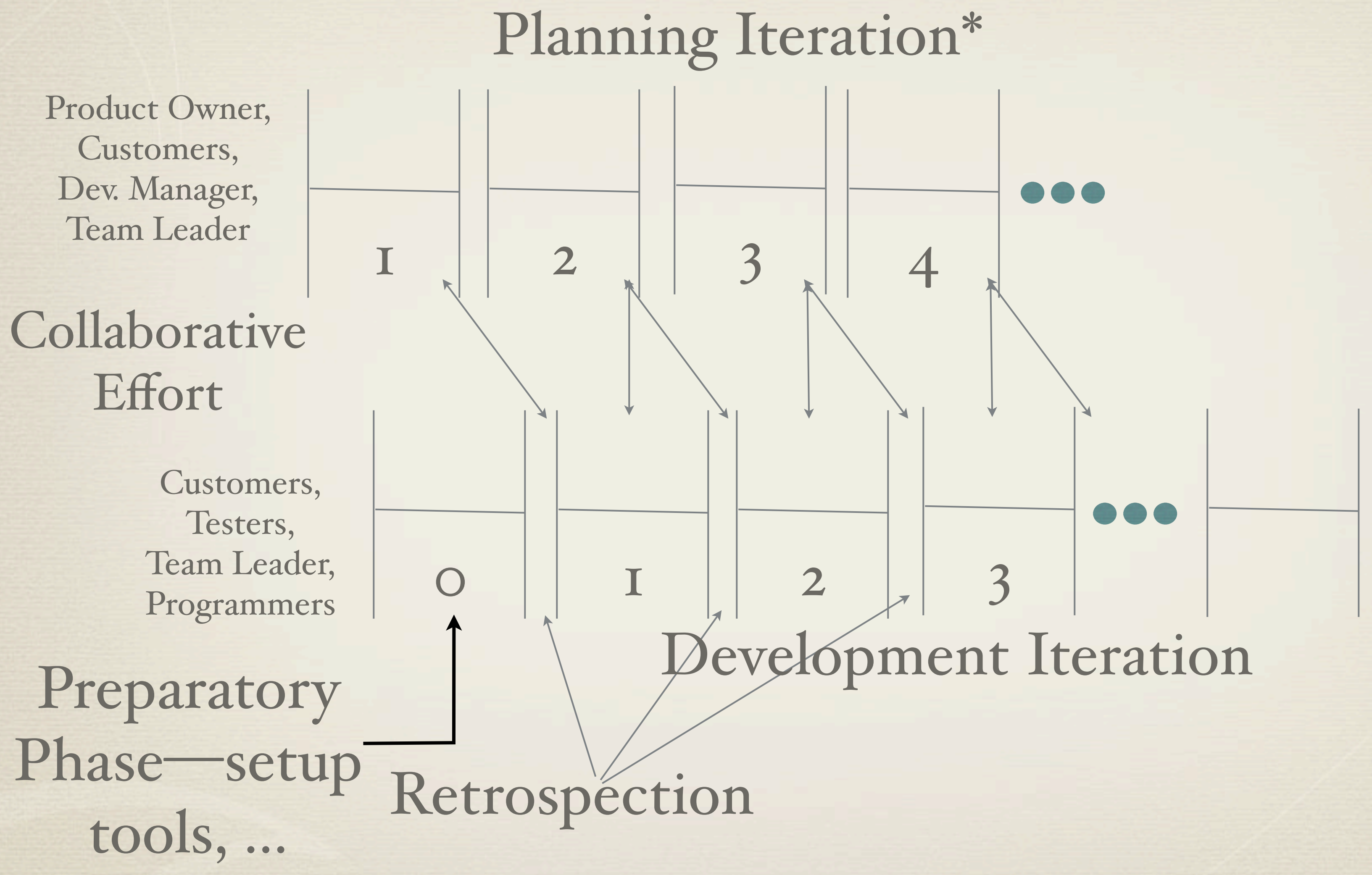
Customer Participation

- Product Owners and Customers need to participate in the development
- They can't be visitors
- They're expected to steer the project
- Active feedback
- They're involved in the adaptive planning

Feedback

- Feedback is critical
- Don’t assume they’ll give it to you
- Solicit Feedback
- What’s worse than not getting feedback? Not doing anything about it.
  - Follow up, tell them what you did or why not

Effective Scrum


Planning Iteration*
Product Owner, Customers, Dev. Manager, Team Leader
1
2
3
4
●●●
Collaborative Effort
Customers, Testers, Team Leader, Programmers
0
1
2
3
●●●
Development Iteration
Preparatory Phase—setup tools, ...
Retrospection

*-Create/Refine Stories, Measure Progress, Re-Prioritize, Exercise App

Iteration Length

- How long?
- It depends
- If you follow Scrum, don’t blindly assume month long
- If you find it useful to follow sub-iterations/sub-sprints, do so

Retrospection

- You don't work for the process
- The process works for you
- Is it working for you?
- What do you like?
- What's not good about it for you?
- Discuss, do more of what's working.
- Address concerns and make changes to what's not working
- If you are on an agile project, you need to fine tune it

How do you know?

- If you ask the team what's working and what's not at the end of iteration, you will hear the words: "It's good" or "It's OK"
- That does not help
- Your pain does not arise at the end of iteration
- It arises everyday
- So, make a note everyday

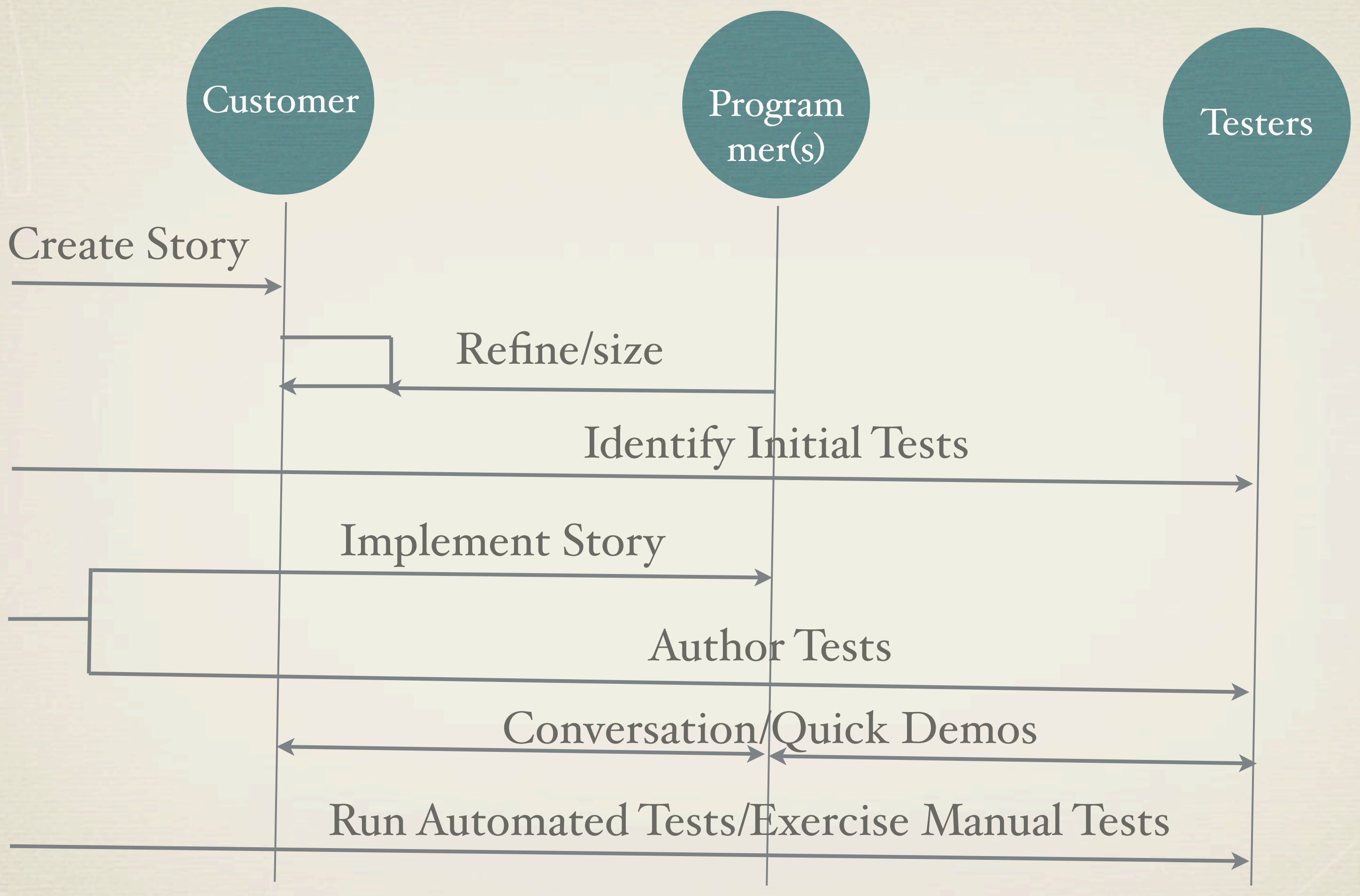
Jot Down

What's working?	What can be better?
…	…
…	…
…	…
…	…
…	…
…	…
…	…
…	…
…	…

Standup Meeting

- Long meetings are counterproductive
- Short meetings to get everyone on the same page
  - What did you do yesterday
  - What's your plan today
  - What's holding you back–blockers
- Not a status meeting
- Identify issues that may need further discussion in smaller groups

Story Progression



Small-Bites

- Set small milestones
- Follow a rhythm
- Collaborate
- Work together, not in isolation
- Keep code in releasable state (for testing)
- Measure progress on a daily basis
- Write code with high quality and good test coverage

Pragmatics

- Ask and *understand* Why?
- Be Adaptive
- Actively Seek Feedback
- Make change affordable and predicatable
- Release Frequently
- Test often and test early
- Automate most of your tests
- Create lean, useful documentation
- Practice Evolutionary Design and Architecture
- Use Unit Testing as safety net for evolutionary design