

Lean Anti-Patterns

An anti-pattern is a commonly occurring activity that is counter-productive to the company doing it. A “Lean” anti-pattern is an anti-pattern that goes against Lean principle. That is, it violates the company’s attempts to be following Lean methods.

The following tables offer a first-cut at commonly occurring Lean anti-patterns. Each table names an anti-pattern, the principles it violates, and the results that are often seen. The anti-patterns are grouped by category showing how the various anti-patterns relate to each other. Often, one anti-pattern contributes to others.

The list is not meant to be complete nor is it inclusive. Treat these tables as the start and the principles and results as illustrations.

We follow eight principles of Lean Software Development, based in part on *Lean Software Development: An Agile Toolkit* (Poppendieck, Mary and Tom):

1. Add value to the customer
2. Eliminate waste
3. Optimize the whole
4. Deliver fast
5. Build quality in
6. Defer commitment
7. Respect people
8. Create knowledge

An alternative view of Lean principles comes from *Lean Thinking* (Womack and Jones):

1. Value
2. Value stream
3. Flow
4. Pull
5. Perfection

Value, Value Stream and Flow are somewhat equivalent to a combination of adding value, eliminating waste, optimizing the whole, delivering fast and building quality in.

Pull and perfection are manifestations of deferring commitment, respecting people and creating knowledge.

Principles inferred from the above include:

1. Limit work to capacity
2. Remove impediments to flow

Lean Anti-Patterns at the Organizational Level

| Anti-Pattern | Principles Violated | Result of Anti-Pattern |
|---|--|--|
| Too many projects | <ul style="list-style-type: none"> Limiting work to capacity Optimizing the whole Eliminate waste | <ul style="list-style-type: none"> Thrashing of the team Decreased efficiencies |
| Not tying the project back to business value | <ul style="list-style-type: none"> Optimizing the whole Eliminate waste | <ul style="list-style-type: none"> Building things that aren't needed |
| Focusing on projects, not products | <ul style="list-style-type: none"> Optimizing the whole Eliminate waste | <ul style="list-style-type: none"> Working on tasks of one project that is of a lower priority than another project. |
| Stove-piping of organization | <ul style="list-style-type: none"> Optimizing the whole Eliminate waste | <ul style="list-style-type: none"> Difficult to create work-cells. Delays increase when need assistance from other stove-pipes |
| Big budgets result in big projects <i>AND</i> Making it easier to get more money once, than the equivalent amount over different projects | <ul style="list-style-type: none"> Eliminate waste Deliver fast Defer commitment | <ul style="list-style-type: none"> Resources spent on parts of projects that are of less value than other projects. Tend to build all of something instead of just the most useful pieces. |
| Multiple, entangled projects with key people being required for all of them | <ul style="list-style-type: none"> Build quality in Optimizing the whole | <ul style="list-style-type: none"> Key people cause delays Teams thrash Coupling between projects causes problems |
| Not having a separate value stream for improving and managing the product value streams | <ul style="list-style-type: none"> Optimize the whole Build quality in Eliminate waste | <ul style="list-style-type: none"> Challenges remain and are hard to eliminate |

Some notes about violating lean principles at the organizational level:

At the organizational level how different aspects of the organization interact are often over-looked. Projects get started and continue through budget even though business value return has decreased.

Lean Anti-Patterns in Project Management

| Anti-Pattern | Principles Violated | Result of Anti-Pattern |
|--|---|--|
| Not building in iterations | <ul style="list-style-type: none"> • Deliver fast • Eliminate waste • Defer commitment | <ul style="list-style-type: none"> • Value delivered late • Features will be built that aren't needed due to lack of feedback from customers |
| Making decisions that are costly to reverse, too early | <ul style="list-style-type: none"> • Defer commitment • Build quality in | <ul style="list-style-type: none"> • Being constrained by decisions that were made with incomplete information |
| Releasing entire system at once when partial releases are possible | <ul style="list-style-type: none"> • Deliver fast • Eliminate waste | <ul style="list-style-type: none"> • Value is delivered late to the client • Feedback is not received quickly • Typically build more than you need |
| Focusing only on project management | <ul style="list-style-type: none"> • Optimize the whole • Build quality in | <ul style="list-style-type: none"> • Companies often work on the wrong part of the development cycle (may involve aspects outside of the development team) • Code quality is ignored |

Some notes about violating lean principles in project management:

Improper project management tends to build the wrong things. This not only wastes time, it adds complexity to the system. An over-focus on project management tends to ignore both design/code quality issues as well as how teams need to interact.

Lean Anti-Patterns in Requirements and Analysis

| Anti-Pattern | Principles Violated | Result of Anti-Pattern |
|--|---|---|
| Unavailable customer | <ul style="list-style-type: none"> • Eliminate waste • Optimize the whole | <ul style="list-style-type: none"> • Will build things that aren't needed |
| No product owner | <ul style="list-style-type: none"> • Eliminate waste • Optimize the whole | <ul style="list-style-type: none"> • Will build things that aren't needed |
| Encouraging customers to specify what they might need instead of just what they certainly need | <ul style="list-style-type: none"> • Eliminate waste • Optimize the whole • Deliver fast | <ul style="list-style-type: none"> • Will build things that aren't needed • Will take longer to build the entire system |
| Too much analysis up-front | <ul style="list-style-type: none"> • Eliminate waste • Optimize the whole • Defer commitment | <ul style="list-style-type: none"> • Delays feedback from customer • Wastes time analyzing things that won't get built |
| Analysis is coupled to implementation | <ul style="list-style-type: none"> • Build quality in • Optimize the whole | <ul style="list-style-type: none"> • Leads to poor designs • Does not take future enhancements into account |

Some notes about violating lean principles in requirements and analysis:

Having the wrong customer representative or allowing them to request more than they need wastes effort and adds complexity to the system.

Impact of Anti-Patterns

| Anti-Pattern | Principles Violated | Result of Anti-Pattern |
|---|---|---|
| Customers changing their minds after we have built the system | <ul style="list-style-type: none">• Eliminate waste• Optimize the whole | <ul style="list-style-type: none">• We'll build features that aren't needed• System complexity goes up |
| Team thrashes | <ul style="list-style-type: none">• Eliminate waste• Optimize the whole• Respect people | <ul style="list-style-type: none">• The development team's efficiency will go down |
| Writing functions that are not used | <ul style="list-style-type: none">• Eliminate waste• Optimize the whole• Build quality in | <ul style="list-style-type: none">• We'll build features that aren't needed• System complexity goes up |

Some notes about the impact of the prior anti-patterns:

These anti-patterns result from the prior ones discussed. However, they are anti-patterns in their own right and are often easier to diagnose.

Lean Anti-Patterns in Design and Coding

| Anti-Pattern | Principles Violated | Result of Anti-Pattern |
|---|---|--|
| Commitment to functionality over code quality | <ul style="list-style-type: none">• Eliminate waste• Optimize the whole• Build quality in | <ul style="list-style-type: none">• Code is tougher and more dangerous to change• Code quality will go down |
| Not setting up a way to keep code independent of each other | <ul style="list-style-type: none">• Eliminate waste• Optimize the whole• Build quality in | <ul style="list-style-type: none">• Complexity will increase• Code will be interconnected, making it more difficult and dangerous to change |
| No unit tests in the code | <ul style="list-style-type: none">• Eliminate waste• Optimize the whole• Build quality in | <ul style="list-style-type: none">• Code will be more dangerous and difficult to change |
| Not refactoring code | <ul style="list-style-type: none">• Eliminate waste• Optimize the whole• Build quality in | <ul style="list-style-type: none">• Code complexity will rise• Code will be more dangerous and difficult to change |

Some notes about violating lean principles in designing and coding:

Building code of high quality takes discipline and a solid knowledge of how to do it. It requires the use of design patterns, refactoring, emergent design and test-driven development. Many teams forget that at the end of the day you are still writing code.

Lean Anti-Patterns in Quality Assurance

| Anti-Pattern | Principles Violated | Result of Anti-Pattern |
|---|---|--|
| Quality assurance looking for bugs instead of improving the process | <ul style="list-style-type: none"> Eliminate waste Optimize the whole Build quality in | <ul style="list-style-type: none"> Code quality will decrease Development process won't improve as much |
| Defining tests late | <ul style="list-style-type: none"> Eliminate waste Optimize the whole | <ul style="list-style-type: none"> Developers misunderstand requirements more often Code quality suffers |
| Optimizing QA | <ul style="list-style-type: none"> Optimize the whole Eliminate waste | <ul style="list-style-type: none"> Testing occurs after coding Developers take longer to write high quality code |

Some notes about violating lean principles in quality assurance

QA needs to work with the development team to improve the development process. The result of not doing this is often a lack of clarity of what is to be built as well as uncertainty as to the quality of what was built.

Impact of Anti-Patterns (redux)

| Anti-Pattern | Principles Violated | Result of Anti-Pattern |
|---|---|--|
| Regression deficit disorder | <ul style="list-style-type: none"> Eliminate waste Optimize the whole Build quality in Deliver fast | <ul style="list-style-type: none"> Regression testing takes more effort than it should Takes longer to test and therefore release Quality of code suffers |
| Poor quality of Code | <ul style="list-style-type: none"> Eliminate waste Optimize the whole | <ul style="list-style-type: none"> Will build things that aren't needed |
| Costs of integration exceeds cost of writing the function | <ul style="list-style-type: none"> Eliminate waste Optimize the whole Build quality in | <ul style="list-style-type: none"> Takes longer to add new functionality than it should Bugs take longer to fix It is often dangerous to change the code |

Some notes about the impact of the prior anti-patterns:

These anti-patterns result from the prior ones discussed. However, they are anti-patterns in their own right and are often easier to diagnose.

Company Profile

Net Objectives provides organizations with an end to end solution for Lean Agile Software Development. Net Objectives has facilitated successful Lean-Agile implementations in both small teams and enterprise-wide organizations in both commercial software products and IT applications. We are active participants in the Lean and Agile community and contribute and incorporate advances in principles, practices and techniques in all that we offer.

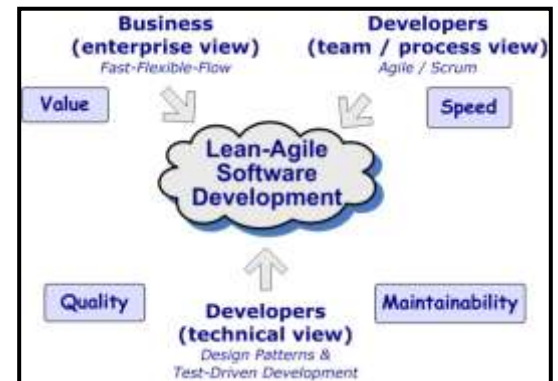
End to End Solution for Lean-Agile Software Development

Organizations looking to deliver the highest value solution to their customers, in stages, in the shortest amount of time, are increasingly interested in Lean Agile Software development.

You can address improvement opportunities for the entire life-cycle of software development, from managing products and projects, to the process of developing software, to the technical skills and techniques needed to build iteratively.

Your company can receive Net Objectives' unique blend of training, intensive coaching and ongoing mentoring, and technical consulting services in the area of software development.

You can be sure of working with the right people to meet your needs; our instructors and senior consultants are experts in Lean-Agile Software Development and have extensive experience in business and software development in the financial, retail, insurance, manufacturing, engineering, and hospitality industries.



Our customers are enabled to deliver solutions in stages in the shortest amount of time. Software development requires an empirical approach where you continually build, inspect, adjust and adapt. Our approach provides you both the short-term knowledge and improvements and the long-term sustained gains that produce continued adoption of the best processes and practices for software delivery.

Your Integrated View: People, Process, and Technical Excellence

The transition to Lean-Agile Software Development impacts many aspects of the organization. A successful transition requires an integrated view: what do the people need; changes to processes; tools and technologies to use. Also, it requires an assessment of impacts on the business, the customers, the team, and project management. It involves driving from the needs of the business while implementing from the reality of the development teams' abilities.

Customers can address all of these areas with our help by focusing on:

- Lean Software Development
- Agile analysis and process
- Architectures that support Agile
- Test-driven development, Design Patterns and Object-Orientation
- Agile project management and facilitation

Contact Mike Shalloway at mike.shalloway@netobjectives.com, 888-LEAN-244, or 404-593-8375 to learn why Net Objectives is the only training/coaching company to offer assistance in Lean, Scrum, Agile Analysis, Design Patterns, Test-Driven Development, QA/FIT and more.