

How lean are you?

If you are not at all familiar with the term "lean", please have a quick look at [Lean Software Development](#).

Look at the following list to get an initial understanding of how lean you are:

- **Amplify Learning:** Do you increase feedback when you have tough problems or do you simply try harder (e.g. by introducing a more disciplined process with more rigorous sequential processing)?
- **Decide as late as possible:** Do you keep your options open as long as practical but no longer?
- **Deliver as fast as possible:** The faster you can deliver, the longer you can delay decisions, e.g. if you can make a software change in a week, then you do not have to decide exactly what you are going to do until a week before the change is needed.
- **Eliminate waste:** Do you spend time only on what adds real customer value?
- **Empower the team:** Do you let people who add value use their full potential or do you command and control?
- **Expertise:** Some knowledge can be codified and shared by documentation, but much knowledge is tacit knowledge that will only be shared through conversation. Do you facilitate a learning environment where tacit knowledge can be shared?
- **Extra features:** Do you implement extra features that don't add real customer value?
- **Extra processes:** Do you have to do time-consuming paperwork that slows down response time, hides quality problems, gets lost, degrades and becomes obsolete?
- **Intuitive decision making:** Do you rely more on your intuition or on your ratio when making decisions? Rational decision making involves decomposing a problem, removing the context, applying analytical techniques, and exposing the process and results for discussion. It suffers from tunnel vision, intentionally ignoring the instincts of experienced people. Rational analysis is unlikely to detect high-stakes mistakes.
- **Iterations:** Iterations are a universal starting point for all agile software development approaches: short feedback loops enhance control; iterations are points of synchronization (the team and the customer see what has been accomplished); iterations force decisions to be made. Do you have short iterations (1 week up to 3 months)?
- **Leadership vs. management:** Do you set direction, align people, and enable motivation? <=> Do you plan and budget the project, organize and staff a project, track and control the execution?
- **Options Thinking:** Premature design commitment restricts learning, exacerbates the impact of defects, limits the usefulness of the product, and increases the cost of change. Do you keep your options open or do you create detailed plans early (carving some decisions into stone)? Plans and predictions are not bad, but making irrevocable decisions based on speculation is to be avoided.
- **Partially done work:** Do you have no partially done work (e.g. requirements specifications for not-yet-implemented requirements, test data for not-yet-executed test cases) or a lot of partially done work?
- **Set-Based Development:** In set-based development, communication is about constraints, not choices. This turns out to be a very powerful form of communication, requiring significantly less data to convey far more information. In addition, talking about constraints instead of choices defers making choices until they have to be made. Do you develop a set of

alternative solutions when you have a difficult problem?

- **Task switching:** Do you have to do a lot of task switching in your daily work?
- **Value Stream Map:** Do you know how much time a typical requirement spends waiting for being processed compared to the time it is being processed?
- **Whole:** Do you see the whole or do you optimize parts at the expense of the whole?