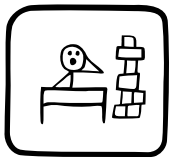
Eight development wastes

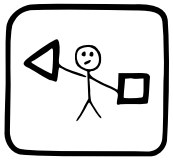
Version 0.4, Copyright © 2011 Mark Seuffert, Sweden



Feature creep

1. Functionality that is rarely used, added without any demand or by wrong assumptions. This happens when features are added that nobody really wants, instead of collaboration with customers and end users.

   What you can do: It's important to know goals, context and priorities in development work.



Inconsistency

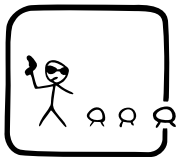
2. Extra work that is caused by doing similar things differently. It takes time to understand how different parts fit together, even more time if they need to be changed or harmonised in order to work well together.

   What you can do: When in doubt aim for consistency, apply the same patterns and standards at all times.

Waiting time

3. Hindrance by waiting times and interruptions. When time is spent waiting for activities and people it prevents more tasks from being finished, in addition has a constant stop-and-go a negative impact on people's productivity.

   What you can do: Visualise work flow, show where bottlenecks and partly finished work occur.



Defect handling

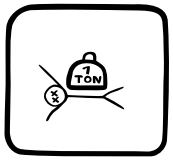
4. Effort for identifying and fixing defects. It's more expensive to catch bugs late in development, at a time when fixing them is harder because details are not familiar any more and problems affect more people.

   What you can do: Instead of mass testing, build quality in right from the start and hold regular design/code reviews.

Information loss

5. Knowledge that is lost by inadequate communication or changing teams. This happens when face-to-face conversation is replaced by documents, important decisions are not written down or only in people's heads.

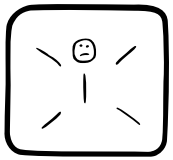
   What you can do: Create easy access to experts and information.



Technical debt

6. Extra work accumulated by development shortcomings. With focus on short term results complexity increases, quality decreases, which causes more maintenance and complete rewrites.

   What you can do: Neglected design is expensive design, spend time on maintainability and refactoring.



Dispersion

7. Hindrance by separation with long feedback loops. Overall productivity drops with specialised or distributed teams, it's better to have crossfunctional teams with collaboration between knowledge domains.

   What you can do: Avoid sub-optimisation and bring passionate people together.

Confining structure

8. Surroundings difficult to change or with insufficient support. Warning signals to look for are heavy management, overly stringent processes, unused talent and unhappy employees.

   What you can do: A healthy system is able to adapt, learn and improve continuously.

Definition: Waste being everything you could remove from a system to get the same (or better) result. Tip: Identify what creates value and what doesn't in the long run, then remove unnecessary or unproductive things from your daily work.

History: The categorisation of waste (Japanese "Muda") has its origin in Lean Manufacturing (Shingo & Ohno) and Lean Software (Poppendiek). The Agile Manifesto knows the principle of maximizing the amount of work not done.