Software Engineering for Sustainability: Find the Leverage Points!

An article by Birgit Penzenstadler, August 2017

By Signe Carlsen
Leverage points (LP)

- Locations within a system where a small change in one aspect can result in significant system-wide changes
- Serve as a tool to help practitioners identify elements that can bring out change at multiple levels
Example: UK public transportation system

- A complex system consists of:
  - Paradigm: set of assumptions: the shared belief that people need transportation
  - Goals: transport people
  - Rules: given by the existing infrastructure
  - Stocks: any quantity: e.g. number of private vehicles
  - Flow: how the level of the stock can be changed
  - Parameters: influence the flow: e.g. congestion fees
  - Feedback loops: occur when a changing level of a stock or a flow creates a circular chain of cause-and-effect that eventually influences the original stock or flow
How can software effect change in its wider environment?

- Meadow’s list of twelve leverage points: categorized into four groups, effectiveness increases
Changing the metabolic structure of the system (LP12, LP11, LP10)

- Change the values of parameters and the material stocks and flows

- Fine-tune the system, without changing its nature: Can help optimize system, but has low leverage

- Software example: by continuously monitoring commuter patterns, software systems can adjust the frequency of buses, to increase the low, or recommend higher capacity buses on certain routes, to reduce overcrowding
Changing the feedback loops (LP9, LP8, LP7)

- Balancing and reinforcing feedback loops

- For example, priority bus lanes can improve the perception of buses as a convenient alternative, and encourage more people to switch to buses, who then demand even more such improvements

- Software can monitor where these priority lanes are most needed
Transformational change (LP6, LP5, LP4)

- Change the structure of information flows and rules of a system
- High leverage as they can sweep away existing feedback loops
- Software example: Uber
- Changes the information flow by connecting people who need transportation with those who can supply it
- Changes the rules by side-stepping existing regulations around licensing and safety regulations for taxis
Change in the intent of systems and stakeholders (LP3, LP2, LP1)

• The most radical system change: the goals or the mindset that creates them (the paradigm)

• Change the goal from being that people choose buses over cars to commute, to removing the need to commute at all

• Software example: help in planning walkable communities with regard to city layout, optimal distribution of living and working areas etc.
Responsibilities as software engineers

- Software systems are deeply ingrained in society and can be drivers of change

- However, we must understand them as drivers of change, e.g. by using LPs, or else we may miss potential

- Our responsibilities are not limited to our customers’ immediate concerns, but include responsibility for the long-term consequences of the systems we design

- Not only ask «Are we building the product right?», but also «Are we building the right product?»