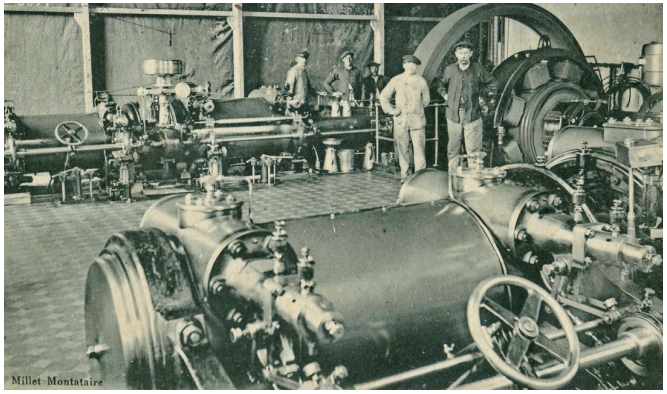
The problem with master-planning

The master-planning paradigm, developed for the factories of the Industrial Revolution, is no longer appropriate as the dominant city-planning paradigm.



It was a rainy night on a highway, two hundred miles from home. My excitement was growing as my brother Stephen explained a revolutionary new approach to software development, which his company had recently adopted. I could see the immediate connections with a new approach to urban design that I had been developing and advocating for several years.

'Traditionally, software development has been based on the waterfall model,' Stephen explained, 'the master-planning process used to design and build machines. With the waterfall model, the design and build goes through a series of steps, like water cascading down a waterfall. When computers were first invented, software developers unconsciously adopted this same waterfall model when designing and building software. But as it turned out, this master-planning approach had some severe limitations.'

The waterfall method of design came out of the Industrial Revolution. It proved extremely useful for designing machines and production lines that operated in static environments that remained unchanged for decades. When the modern town-planning movement began to emerge at the turn of the last century, it also unconsciously adopted this master-planning method of design, just as the software developers did sixty years later. It *assumed* that designing a town or city was similar to designing a better machine. Specialists were trained to look after their part of the machine – urban planners looked after land use, traffic engineers looked after roads and traffic flow, landscape architects looked after parks, etc. Each tried to optimise their part of the machine. For example, traffic engineers would try to optimise the efficiency of an intersection, exactly like a factory engineer would try to optimise the efficiency of a production line – by focusing on increasing through-put.

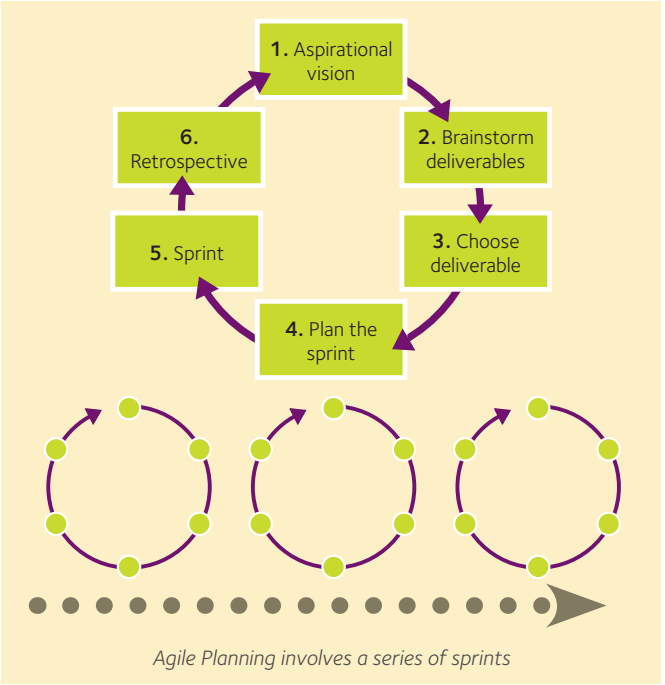
As we continued our journey on the rain-soaked highway, Stephen explained: 'The waterfall model works fine for designing better machines which operate in a static world. But software operates in a dynamic, ever-changing world, where technological revolutions are a daily occurrence. The waterfall model does not have the flexibility to deal with unknown challenges and unknown opportunities. For dynamic systems you need a more agile approach.'

This has been the 'elephant' in Council and government offices worldwide. While some elements of towns and cities have a machine-like quality (just like a human arm operates as a lever), urban environments are dynamic, living, fast-changing environments. The master-planning paradigm, developed for the factories of the Industrial Revolution, is no longer appropriate as the dominant paradigm for planning our cities.

Here are the fatal flaws for software – you make the connection to the way we try to create better public places (OK, I will give you some hints).

- Software takes several years to develop, by which time the computer world has changed dramatically and the software is outdated – so a lot of very expensive software (some costing millions of dollars to develop) ends up gathering dust on a shelf. (Think of all the very expensive ten-year plans and grand plans for great public spaces sitting on Council shelves gathering dust.)
- Small mistakes made at the start of the development process become entrenched and built into the software, making it difficult and expensive to remove these bugs at the end of the development process. (I have done hundreds of audits of public places and am constantly surprised by how one small oversight at the start of the design process becomes a 'bug' that severely limits the functionality of the space.)
- The clients only know about 20% of what they really want from the software at the start of the process. They discover the other 80% during the development process. Because the analysis-of-requirements phase is at the start of the process, and has long passed, the resultant software is only 20% as effective as it could have been. (Think of the very expensive makeovers that seem to have delivered only 20% of their potential. And think of all the wasted community consultation because we assumed the community knew what it wanted, when all it could imagine is what it had already seen some-place-else.)

E-book: creative-communities.com/learning-centre/



Agile Planning involves a series of sprints