http://youtu.be/GE6lbPLEAzc (43 min)

…or see similar talk: http://www.youtube.com/watch?v=TgdFA72crHM (fast rewind to 25:30 and look forward, total 25 min)

Explaining Agile

Martin Fowler and Neil Ford at USI

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

http://agilemanifesto.org

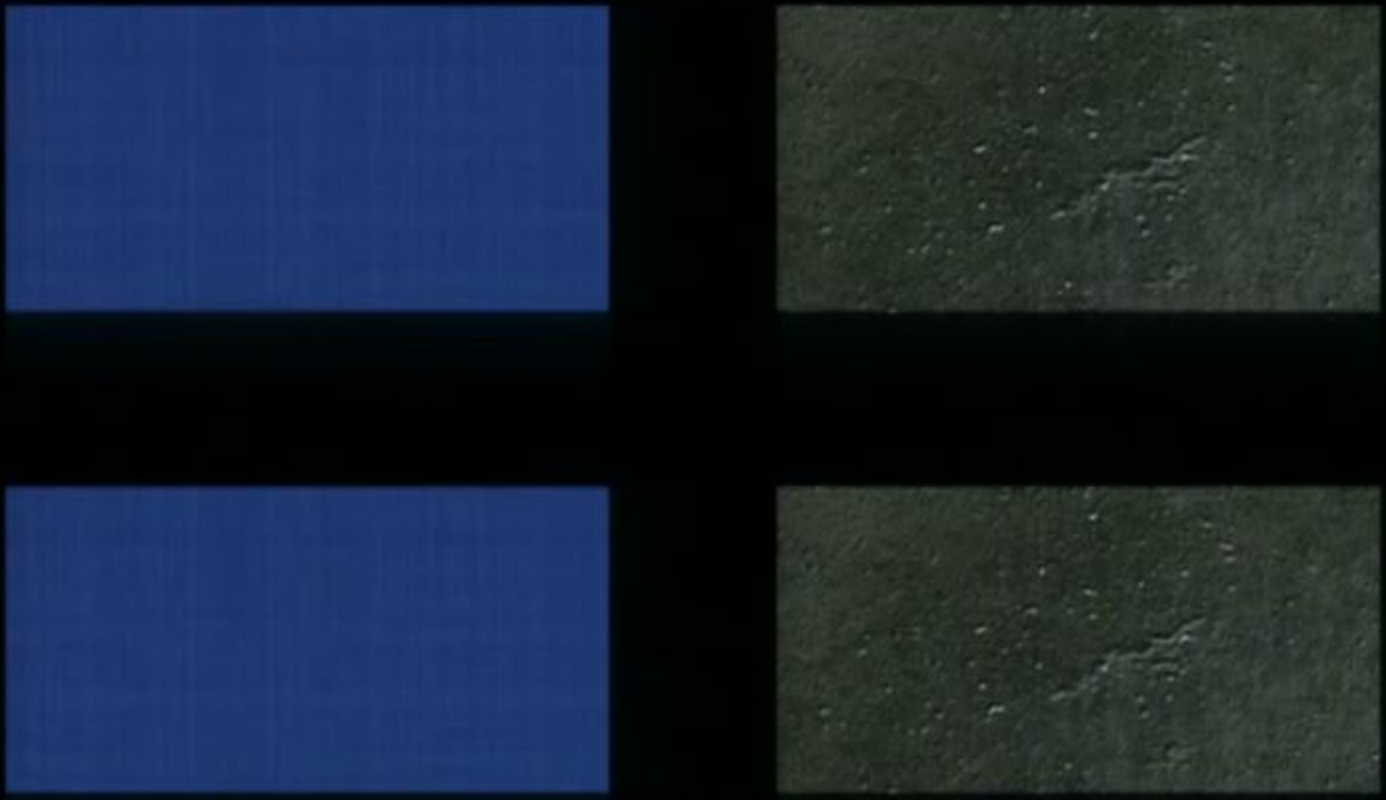
The original ideas of what agile is has often passed from person to person and has gotten more and more diffuse…

…and as a result we often enters organizations where they "are doing agile"…

…but when we look at what they are doing there is nothing that we can recognize as an agile approach.

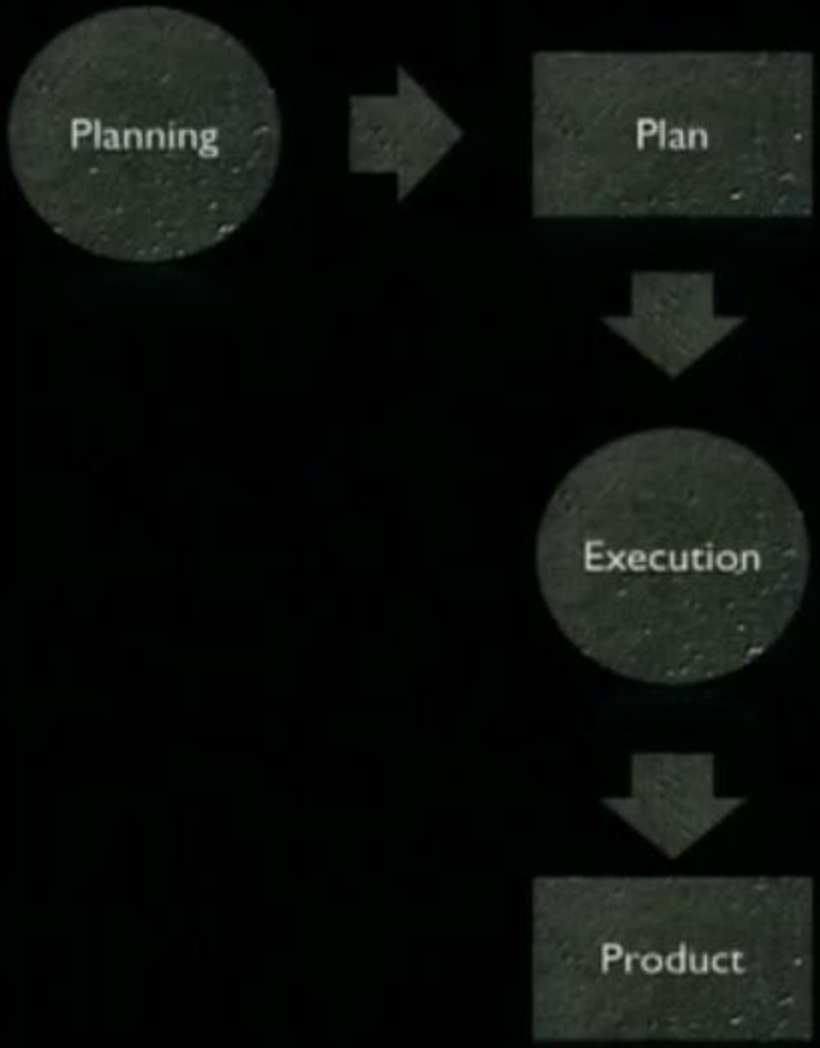
Agile

Plan-Driven



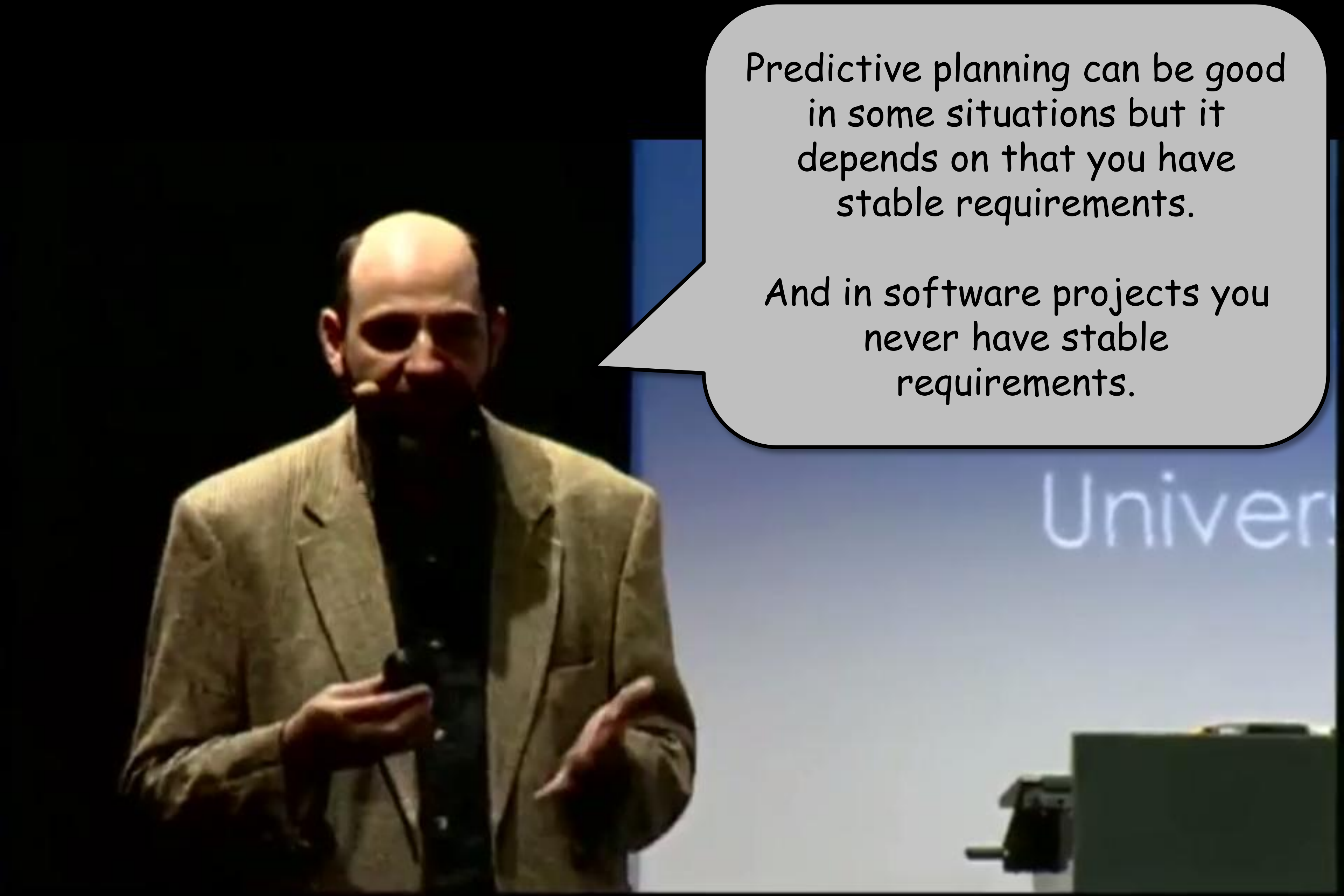
http://martinfowler.com/articles/newMethodology.html

Predictive Planning



Predictive
Planning

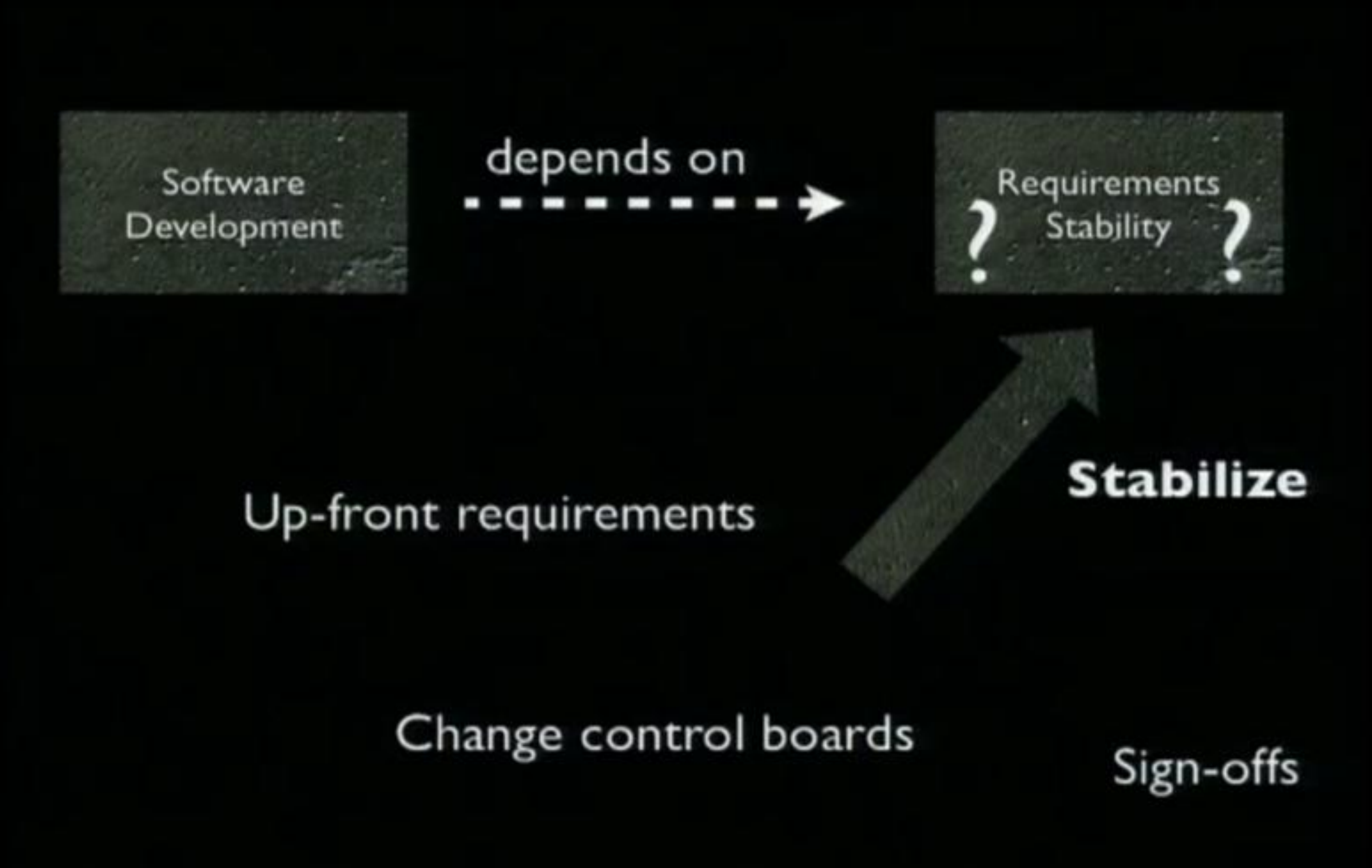
success == according to plan



Predictive planning can be good in some situations but it depends on that you have stable requirements.

And in software projects you never have stable requirements.

The plan-driven community often try to force stabilization of the requirements. And they come up with all kind of techniques to do that…



Software Development

depends on

Requirements Stability

? ?

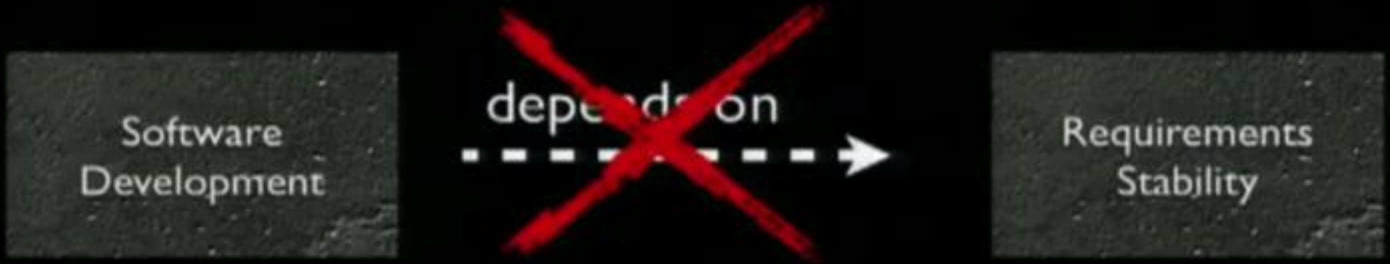
Stabilize

Up-front requirements

Change control boards

Sign-offs

…but often one find that this is really kind of hard.



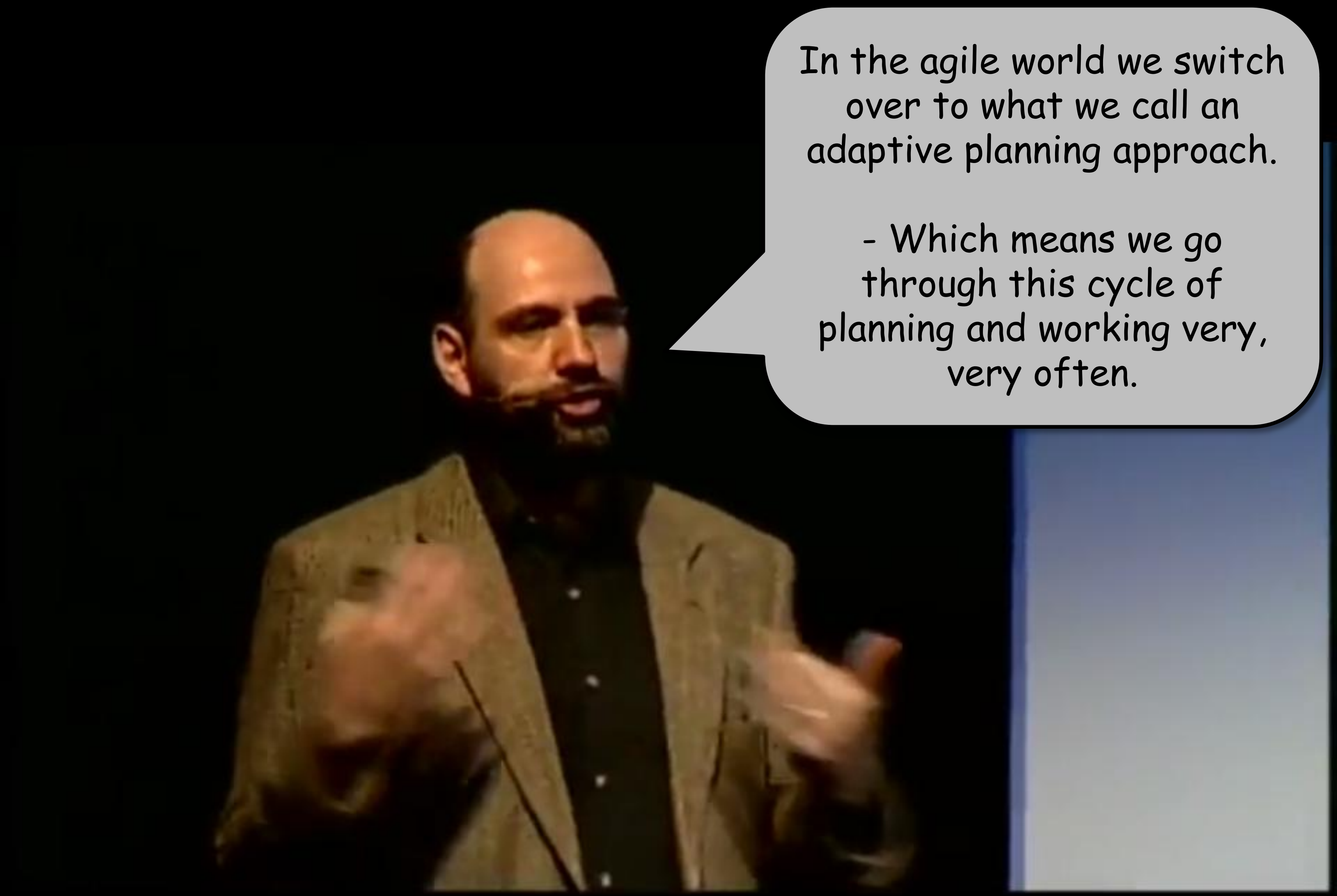
Software Development

depends on

Requirements Stability

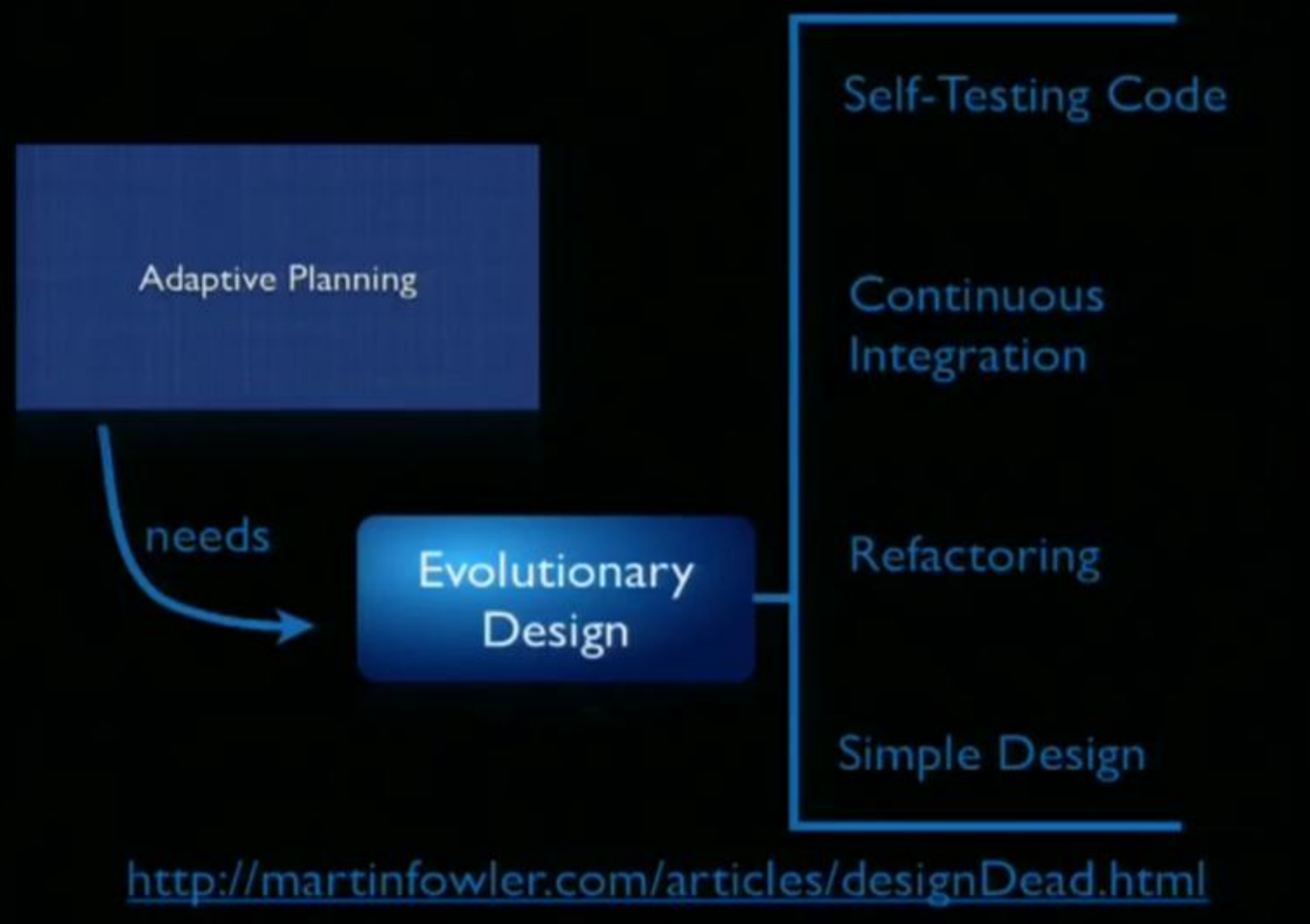
"A late change in requirements is a competitive advantage"

Mary Poppendieck


In the agile world we switch over to what we call an adaptive planning approach.
- Which means we go through this cycle of planning and working very, very often.

Agile	Plan-Driven
Adaptive Planning	Predictive Planning

These technical areas must be in place for being agile in software development.



Adaptive Planning

needs

Evolutionary Design

Self-Testing Code

Continuous Integration

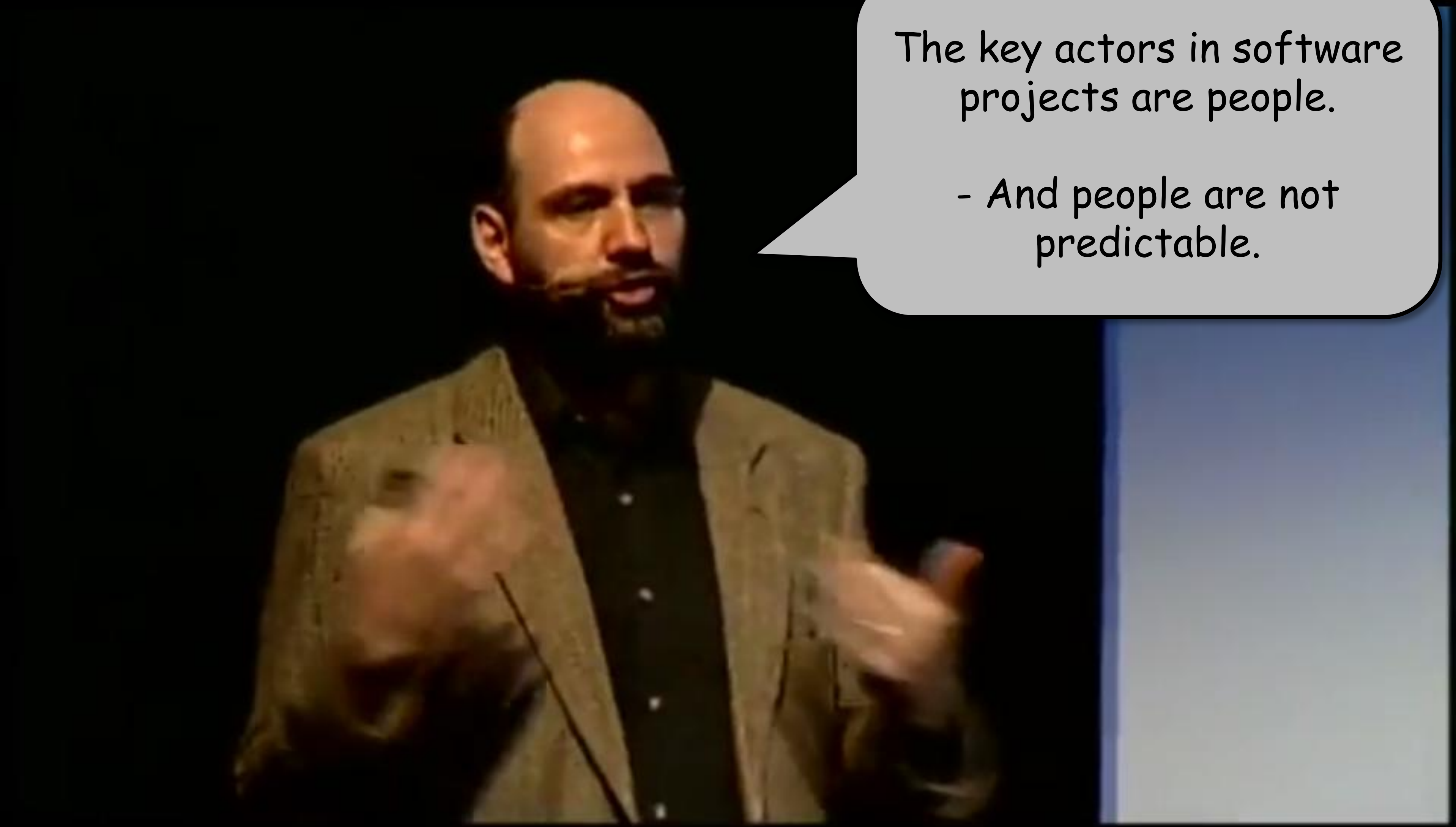
Refactoring

Simple Design

http://martinfowler.com/articles/designDead.html

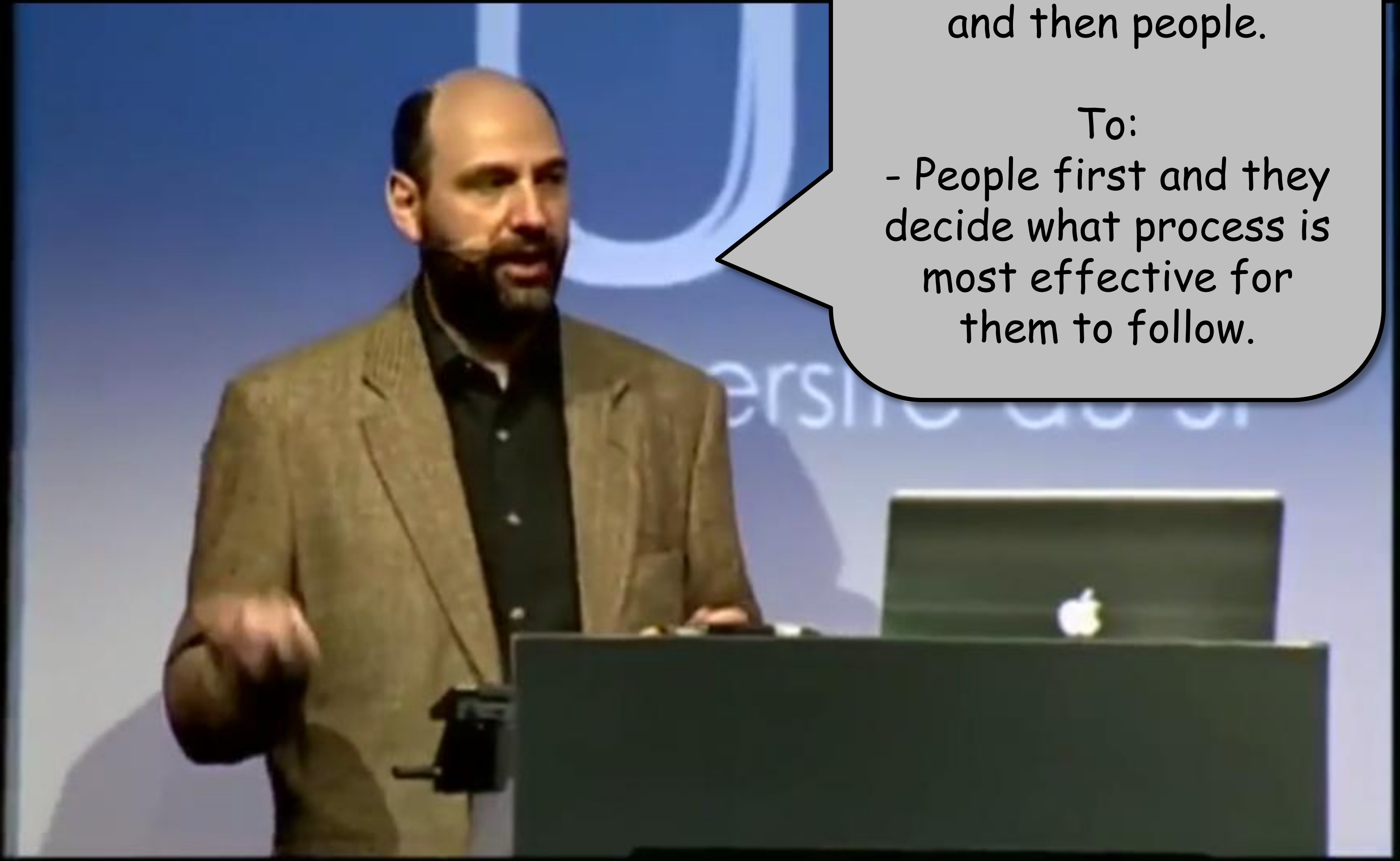
In the title, [of his article] I refer to people as "components". That is how people are treated in the process / methodology design literature. The mistake in this approach is that "people" are highly variable and non-linear, with unique success and failure modes. Those factors are first-order, not negligible factors. Failure of process and methodology designers to account for them contributes to the sorts of unplanned project trajectories we so often see.

Alistair Cockburn: *Characterizing People as Non-Linear, First-Order Components in Software Development*



The key actors in software projects are people.

- And people are not predictable.



Shift from:
- Process comes first and then people.

To:
- People first and they decide what process is most effective for them to follow.

A bad process will beat a good person every time

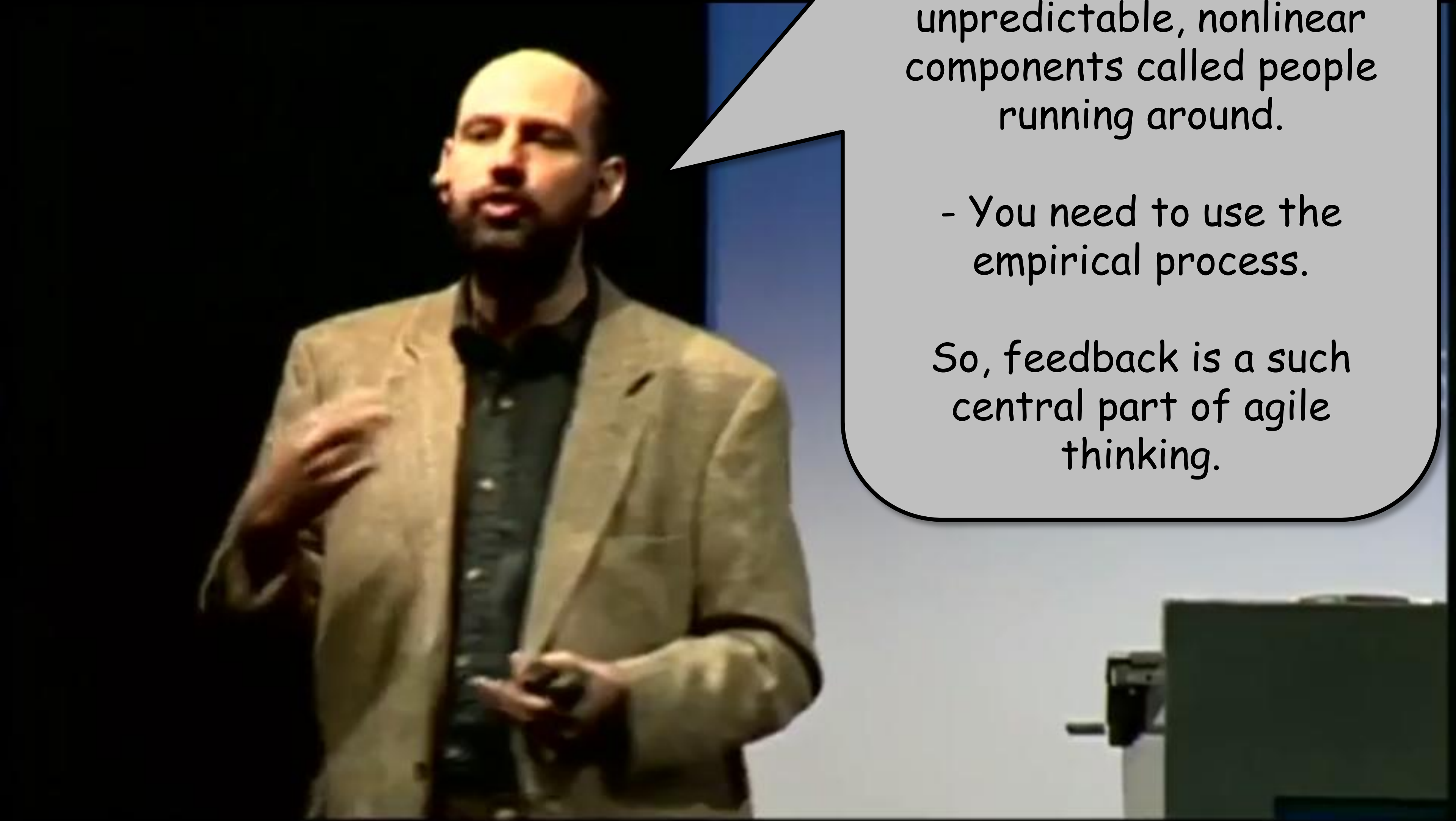
W. Edwards Deming (attr)

Agile	Plan-Driven
Adaptive Planning	Predictive Planning
People-first	Process-first

Empirical Process

source: Ken Schwaber interview in Agile Software Development Ecosystems by Jim Highsmith

Defined Process



In software project we have all these unpredictable, nonlinear components called people running around.

- You need to use the empirical process.

So, feedback is a such central part of agile thinking.