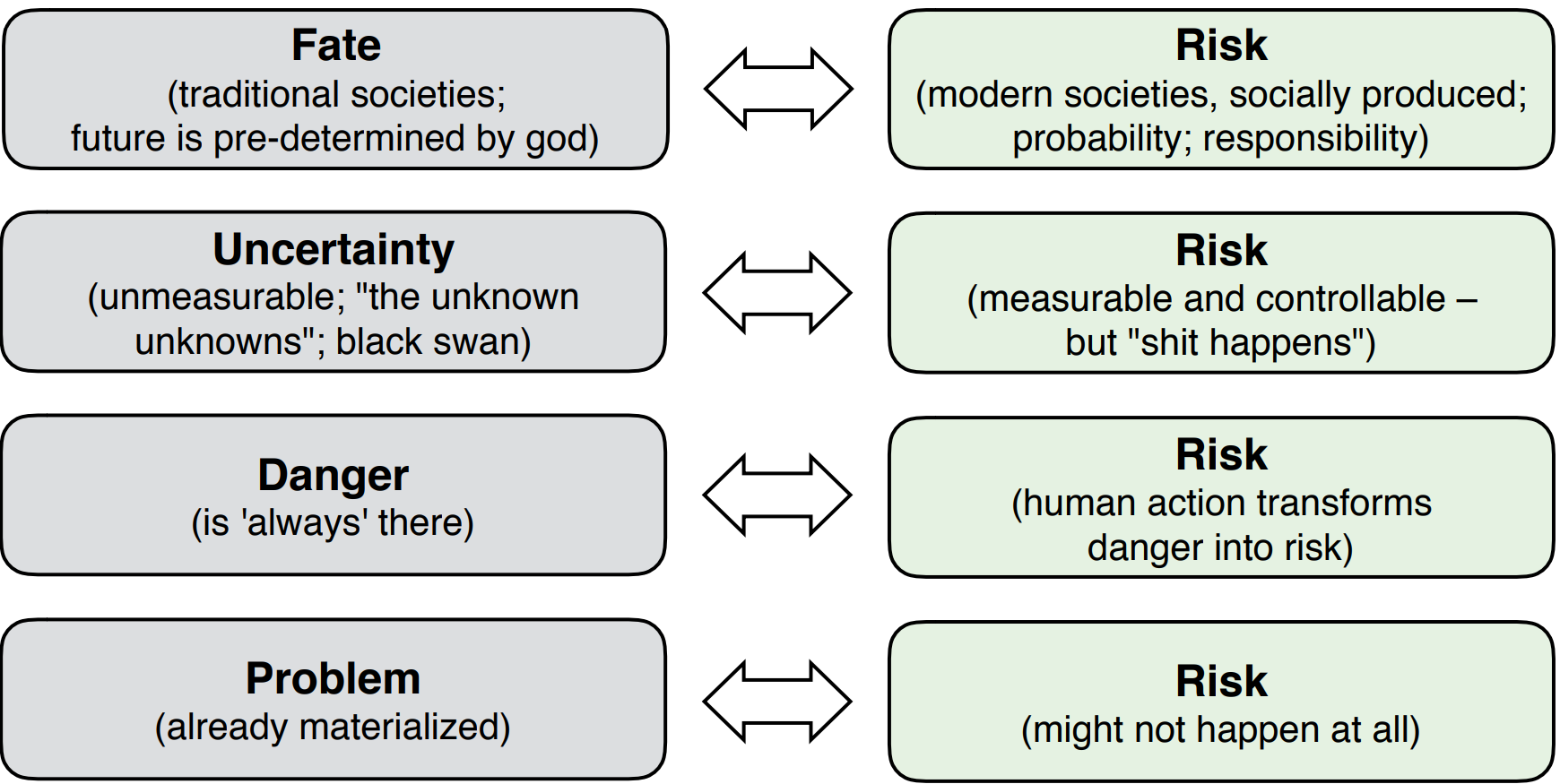
**POM 17 Risk Management**

**Risk:**

* is the deviation of a result of a future event from expectations
* is uncertain whether it will materialize or not
* is a potential harm that may arise of such an event
* depends on our decisions

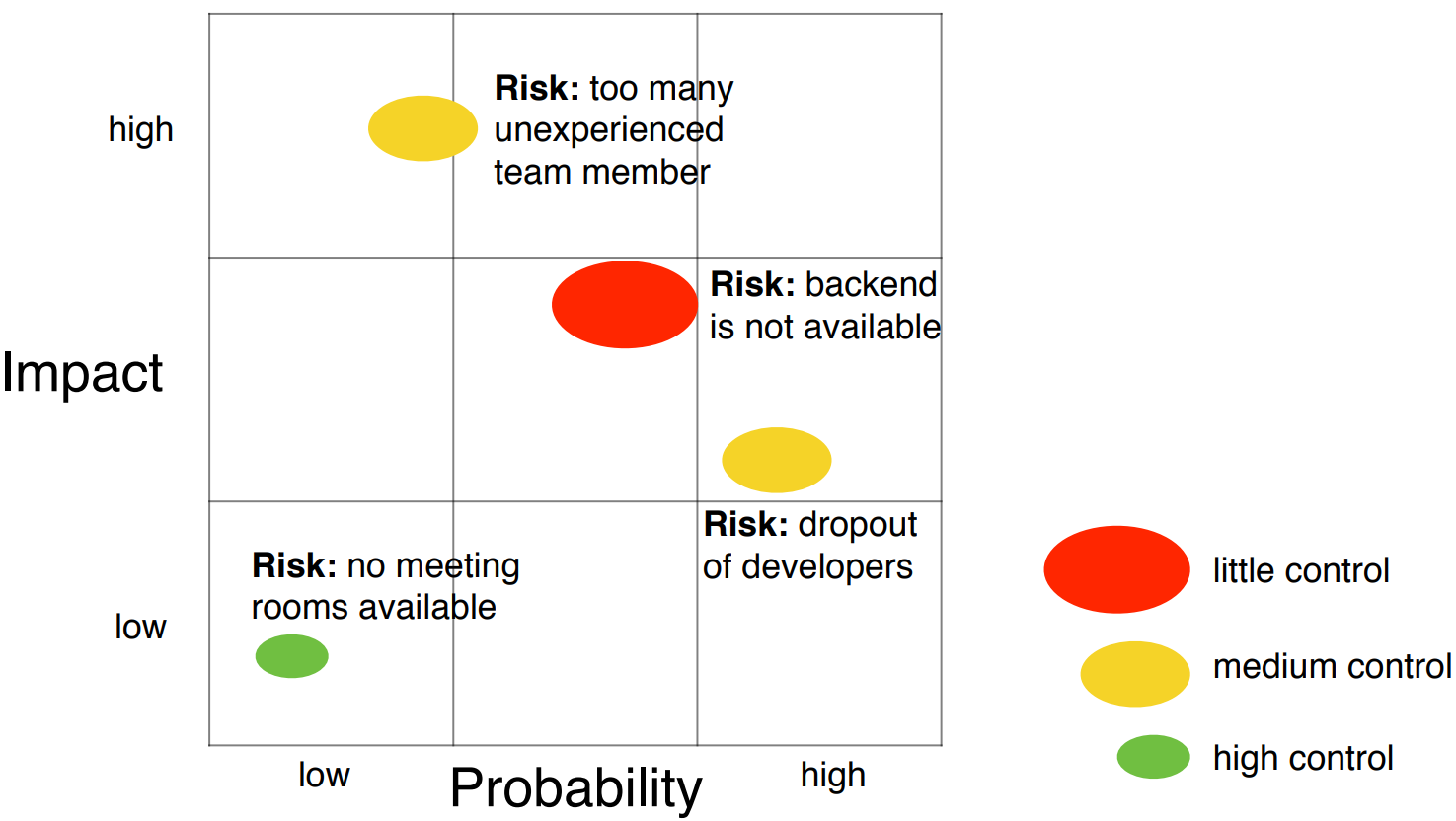
Risk Differentiation:



Identify Risks:

* What might affect project objectives?
* Internal or external sources?
* Performed by project management AND team #
* Methods: brainstorming, checklists, expert discussions, SWOT analysis (strengths, weaknesses, opportunities, threats)
* Group risks into categories (e.g. complexity, technology, resources, quality, requirements, planning, communication & change management)
* Assign risk triggers
* Examples in agile projects
  + Daily scrum meeting: status, impediments, promises
  + Sprint review meeting, sprint retrospective: identify improvements for future sprints

Risk assessment (bewerten) matrix



**Murphy’s Law**: “Anything that can go wrong, will go wrong”

**Risk mitigation**(Minderung):

1. Avoidance

* Try to avoid risk occurrence
* Mostly used at planning phase

1. Control

* Identify actions to control risk occurrence and impact

1. Transfer

* Delegate risk, for example to sub-contractors

1. Investigation

* Try to analyse risk further

**Motivation**

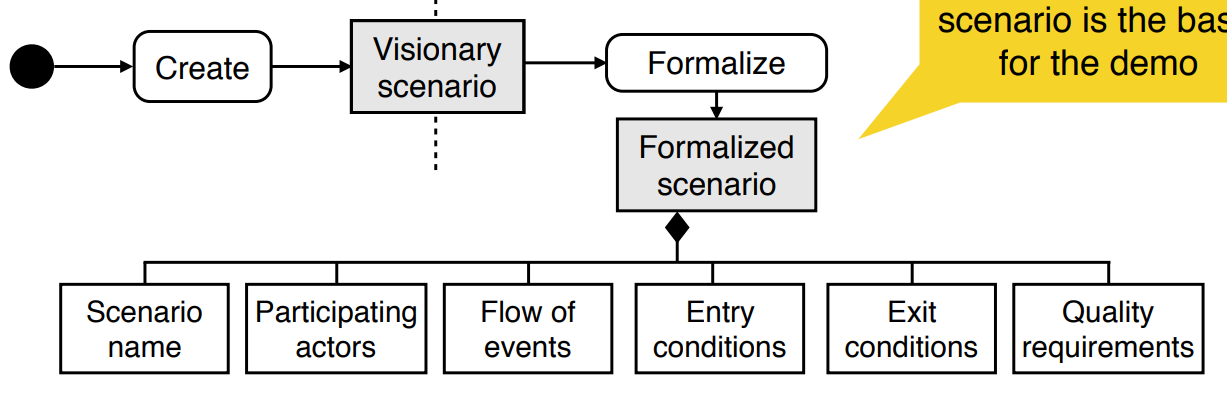
* Boring demonstrations do not convince your customers for future investments or your user to buy the software
* Combination of technical and marketing skills required

How to make a great, convincing and entertaining demo?

* Selection of the right demo: focus on the core value to end users
* Influenced by the real world usage of the software
* Described by visionary scenarios in the problem statement

**Tornado model**: from visionary scenarios (Tornado ganz oben) to a demo scenario (Spitze)

Formalized scenario + 6 Components:



**Screenplay**

consists of

* Scenes describing the event flow and participating actors of the formalized scenario which are needed for the demo scenario
* Additional information such as props and stage directions

**Model-based demonstration**

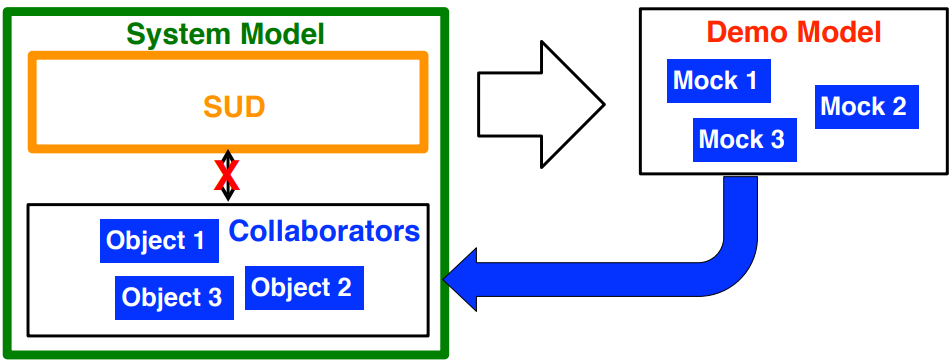
1) We start with the System Model

2) We identify the System Under Demo, SUD

3) The SUD does not exist in isolation, it interacts with other objects that are not yet implemented: Collaborators

4) To be able to interact with Collaborators, we create Mocks

5) Mocks mimic the behavior of the Collaborators



**Mock object pattern**

* The mock collaborator replaces the behavior of the collaborator (the real object)
* A mock object can be created at startuptime with a factory pattern
* Mock objects can be used for testing state of individual objects as well as the interaction between objects
* We also use mocks for the demo

