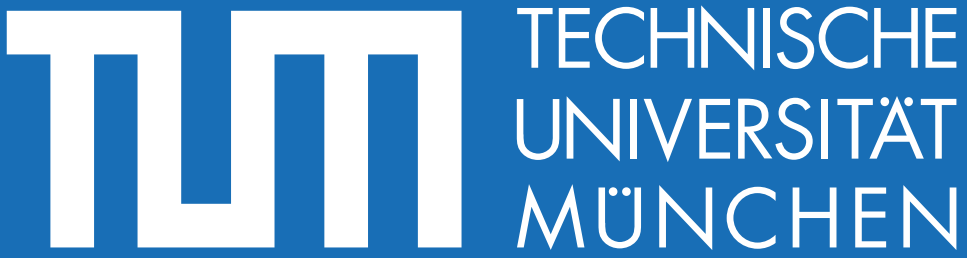



# Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall

ENGINEERING FUNDAMENTALS

# Sustainability

TECHNISCHE UNIVERSITÄT MÜNCHEN




Nachhaltiges Handeln sollte selbstverständlich sein, da es immer andere nach uns geben wird. Im Software Engineering bieten sich die folgenden Minimierungs-Prinzipien an, um nachhaltig zu agieren:

Minimierung des Leerlaufs und Maximierung der Nutzung der vorhandenen Hardware-Ressourcen; Minimierung der Ineffizienz und Maximierung der Effizienz von Software-Anwendungen und deren Entwicklungsprozessen; Minimierung der Gesamtmenge der gesammelten, übertragenen, gespeicherten und verbreiteten Informationen; Minimierung der Exzessivität und Maximierung der Angemessenheit von Lösungsdesigns; Minimierung der Vergänglichkeit und Maximierung der Lebensdauer jeglicher Art von Lösungen; Minimierung der Ausbeutung und Maximierung des Rückflusses in allen Arten von Ökosystemen; Minimierung des Einsatzes von Menschen und Maximierung des Einsatzes von Maschinen in allen Produktions- und Betriebsprozessen; Minimierung des Verbrauchs und Maximierung der Einsparung von Energie in allen Produktions- und Betriebsprozessen; und: Minimierung der Kohlenstoffemissionen und somit des Fußabdrucks bei allen Produktions- und Betriebsprozessen.

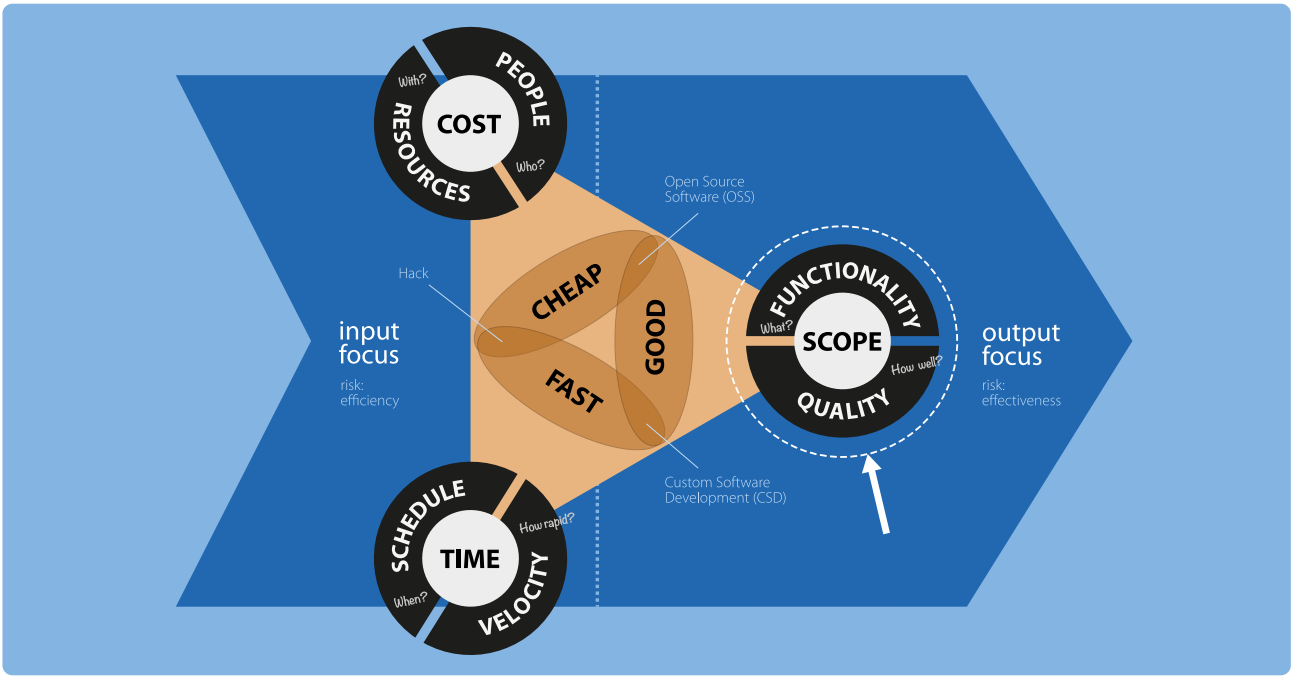
## Fragen

- Ist die bewährte Praktik des Continuous Integration (CI) ein nachhaltiges Handeln?

# Project Management Triangle








**Definition of a *Project*:**
*"Temporary endeavor undertaken to create a unique product, service or result."*
**Temporary** in that it has a defined beginning and end in time, and a defined scope and cost.
**Unique** in that it is not a routine operation, but a one-time, single-goal, and risk-containing operation.

**Project Management *Iron Triangle*:**
A project is constrained by **time**, **cost** and **scope**. No constraint in this triangle can be changed without affecting the others.
Time splits into **schedule** and **velocity**.
Cost splits into **people** and **resources**. Scope splits into **functionality** and result **quality**.

**Project Management *Trilemma*:**
*"Fast. Cheap. Good. Pick two!"*
Each project optimization effort has the choice among **three** favourable options — only **two** of them are possible at the same time.



**Project Management** ist neben **Software Architecture** die zweite wichtige Disziplin im Bereich **Software Engineering**. Deshalb sollte jeder zumindest ein Grundverständnis über die wesentliche Aufgabe des Project Management haben: die Balance aus dem "Iron Triangle" aus **Time** (Termine), **Cost** (Kosten) und **Scope** (in diesem Kontext üblicherweise Leistung genannt) kontinuierlich zu finden und zu halten.

Die "Stellschraube" **Time** unterteilt sich in die beiden Aspekte **Schedule** (When?) und **Velocity** (How rapid?). Die Stellschraube **Cost** unterteilt sich in die beiden Aspekte **People** (Who?) und **Resources** (With?). Die Stellschraube **Scope** unterteilt sich in die beiden Aspekte **Functionality** (What?) und **Quality** (How well?).

Wenn an einer der drei Stellschrauben bzw. an einer der sechs Aspekte eine Änderung stattfindet, ist das "Iron Triangle" außer Balance und man muss unweigerlich eine oder mehrere der anderen Stellschrauben bzw. Aspekte verändern, um die Balance wieder herzustellen.

Ebenfalls erwähnenswert ist das **Trilemma**, das besagt, daß man üblicherweise immer nur zwei von drei Dingen gleichzeitig haben kann: entweder billig und gut (Open Source Software), dann aber nicht schnell; oder gut und schnell (Custom Software Development), dann aber nicht billig; oder schnell und billig (der "Quick Hack"), dann aber nicht gut.

Die Nicht-Projekt-Manager sind in der Praxis vor allem bei der Stellschraube **Scope** in der Mitverantwortung, da hier eine Veränderung im Projekt üblicherweise ein tiefgreifenderes technisches Verständnis der Anwendung bedingt.

## Fragen

- An welcher "Stellschraube" des **Project Management** sind in der Praxis auch die nicht-Projekt-Manager stark mitverantwortlich?