



# **Software Engineering in der industriellen Praxis (SEIP)**

**Dr. Ralf S. Engelschall**

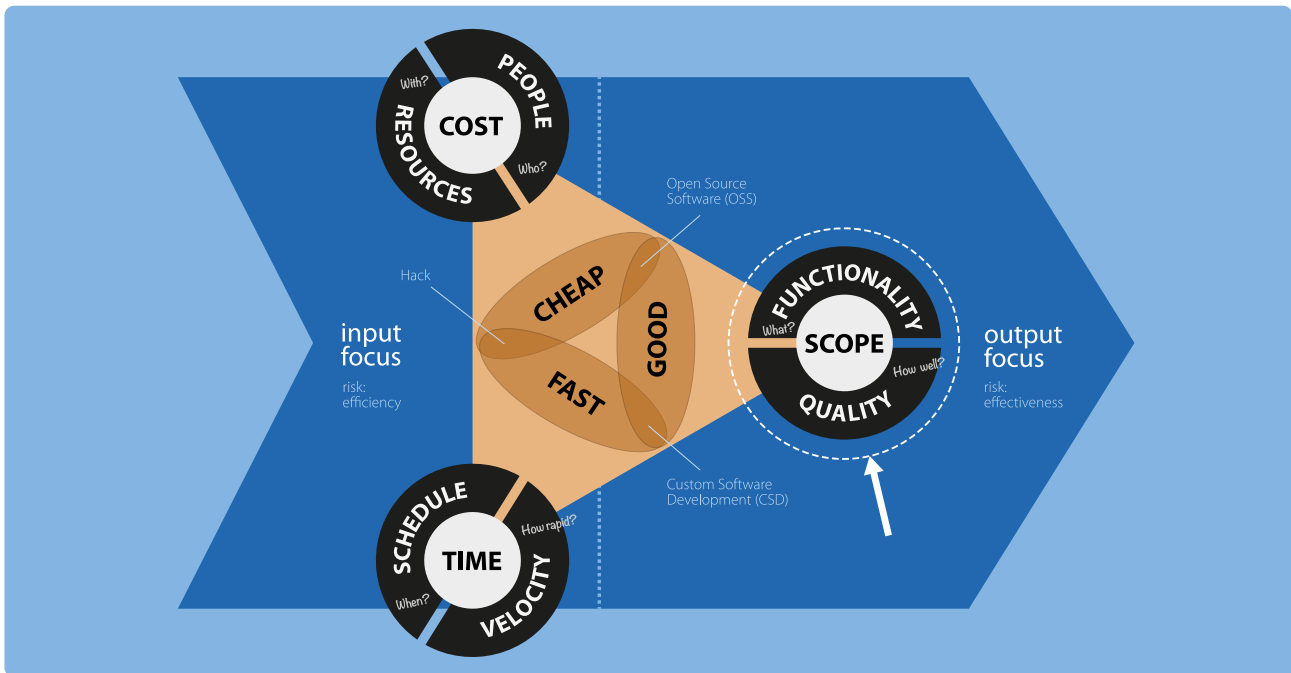
<p><b>HI</b> Minimize <b>HARDWARE</b> Idleness</p> <p>Minimize the idleness and maximize the utilization of existing hardware resources.</p> <p><b>Rationale:</b> Unused or under-utilized hardware are an unnecessary waste of already available resources.</p> <p><b>Keywords:</b> Virtualization, Utilization.</p> 	<p><b>DE</b> Minimize <b>DESIGN</b> Excessiveness</p> <p>Minimize the excessiveness and maximize the adequacy of solution designs.</p> <p><b>Rationale:</b> Non-adequate designs cause unnecessary complexity and waste resources.</p> <p><b>Keywords:</b> Reduced Libraries, Immutability.</p> 	<p><b>HE</b> Minimize <b>HUMAN</b> Effort</p> <p>Minimize the efforts of humans and maximize the efforts of machines in all production and operation processes.</p> <p><b>Rationale:</b> Delegating tasks to machines gives humans the possibility to concentrate on more important tasks.</p> <p><b>Keywords:</b> Computer, Robot, Automation.</p> 
<p><b>SI</b> Minimize <b>SOFTWARE</b> Inefficiency</p> <p>Minimize the inefficiency and maximize the efficiency of software applications and their development processes.</p> <p><b>Rationale:</b> Efficient software and development processes consume less resources.</p> <p><b>Keywords:</b> Caching, Monolith.</p> 	<p><b>SE</b> Minimize <b>SOLUTION</b> Ephemerality</p> <p>Minimize the ephemerality and maximize the life-span of any type of solutions.</p> <p><b>Rationale:</b> Short life-spans of solutions cause unnecessary short renewals and this way wastes resources.</p> <p><b>Keywords:</b> High Quality, Best Practice.</p> 	<p><b>EC</b> Minimize <b>ENERGY</b> Consumption</p> <p>Minimize the consumption and maximize the saving of energy in all production and operation processes.</p> <p><b>Rationale:</b> Electric energy still has to be partially generated from non-renewable resources.</p> <p><b>Keywords:</b> Eco Mode, Reduced CI/CD.</p> 
<p><b>IA</b> Minimize <b>INFORMATION</b> Amount</p> <p>Minimize the total amount of gathered, transmitted, stored and spreaded information.</p> <p><b>Rationale:</b> Reduced amount of information means less data transmission, less data storage, less GDPR issues, etc.</p> <p><b>Keywords:</b> Compression, No Big Data.</p> 	<p><b>EE</b> Minimize <b>ECOSYSTEM</b> Exploitation</p> <p>Minimize the exploitation and maximize the back-contribution in any type of ecosystems.</p> <p><b>Rationale:</b> The consumer and provider behaviour have to be in balance for every long-lasting ecosystem.</p> <p><b>Keywords:</b> Open Source Software.</p> 	<p><b>CE</b> Minimize <b>CARBON</b> Emission</p> <p>Minimize the carbon emission and hence the footprint during any type of production and operation processes.</p> <p><b>Rationale:</b> Climate change and global warming is partially caused or at least accelerated by carbon emissions.</p> <p><b>Keywords:</b> Reduced CO2 Footprint.</p> 

Sustainable action should be a matter of course, since there will always be others coming after us. In Software Engineering, the following minimization principles lend themselves to acting sustainably:

Minimize the idleness and maximize the utilization of existing hardware resources; Minimize the inefficiency and maximize the efficiency of software applications and their development processes; Minimize the total amount of gathered, transmitted, stored and spreaded information; Minimize the excessiveness and maximize the adequacy of solution designs; Minimize the ephemerality and maximize the life-span of any type of solutions; Minimize the exploitation and maximize the back-contribution in any type of ecosystems; Minimize the efforts of humans and maximize the efforts of machines in all production and operation processes; Minimize the consumption and maximize the saving of energy in all production and operation processes; and: Minimize the carbon emission and hence the footprint during any type of production and operation processes.

## Questions

- ❓ Is the Best Practice of Continuous Integration (CI) a sustainable way of acting?



#### 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 & constraints

**Project Management**, alongside **Software Architecture**, is the second important Discipline in the field of **Software Engineering**. Therefore everyone should have at least a basic understanding of the essential task of Project Management: continuously finding the balance from the "Iron Triangle" of **Time**, **Cost** and **Scope**.

The adjusting screw **Time** is divided into the two aspects **Schedule** (When?) and **Velocity** (How rapid?). The adjusting screw **Cost** is divided into the two aspects **People** (Who?) and **Resources**. (With?). The adjusting screw **Scope** is divided into the two aspects **Functionality** (What?) and **Quality** (How well?).

If a change is made to one of the three adjusting screws or one of the six aspects, the "Iron Triangle" will be unbalanced, and one must inevitably change one or more of the other screws or aspects to restore the balance.

Also worthy of mention is the **Trilemma**, which says that one can usually have only two out of three things at a time: either cheap and good (Open Source Software), but not fast; or good and fast (Custom Software Development), but then not cheap; or fast and cheap (the "Quick Hack"), but then not good.

In practice, the non-Project-Managers are co-responsible, especially in the area **Scope**, since here a change in the project usually requires a deeper technical understanding of the Application.

## Questions

- ? At which adjusting skew of **Project Management** in practice are the non-Project-Managers co-responsible?