Swystems
Ankengasse 3
CH-8902 Urdorf

SESE Tour 2022

Sustaining Sustainable Systems

MARCO CHICHERIO
MARCO.CHICHERIO@SWYSTEMS.CH
SWYSTEMS GMBH

Agenda

- Introduction
- What is not sustainable?
- Configuration management
- Requirements – a formal definition
- Classification of requirement changes
- Impact on configuration management
- Conclusion & discussion

Swystems

About the presenter

- Marco Chicherio
  Dipl El-Ing ETH / MSc ITET / E-MBA HEC UNIL

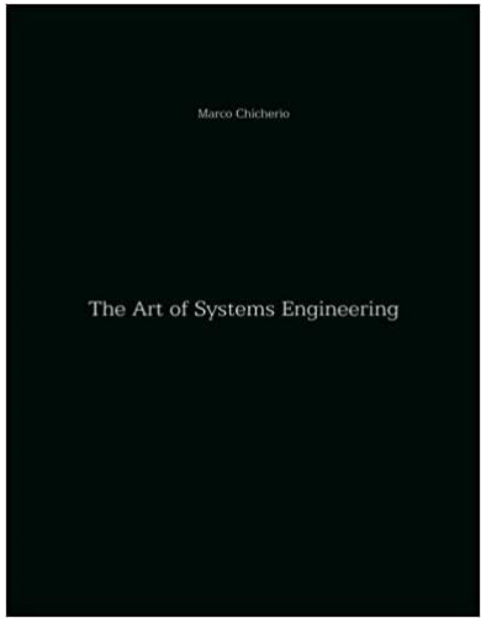
- 2004 – 2006 Horsch Elektronik AG, Engineer / Project Manager
- 2006 – 2011 Thales Suisse SA, Senior Systems Engineer / Systems Architect
- 2011 – 2012 Siemens (DK), Principal Test and Integration Engineer
- 2012 – 2020 Thales Suisse SA
  - 2015 - 2016 Bid Manager / Technical Manager BODLUV 2020 MR
  - 2018 - 2020 Bid Director
- since 2020 Geschäftsführer Swystems GmbH

Swystems

About the presenter

- Marco Chicherio
  Dipl El-Ing ETH / MSc ITET / E-MBA HEC UNIL

- The Art of Systems Engineering
  M. Chicherio, 2017
  ISBN 9 781546 424949



What is not sustainable?

A sustainable design is not sustainable

if operators are not able to sustain it

What is not sustainable?

- Having to replace a system after a few years
- Having to replace a system because it cannot be maintained and sustained
- Not being allowed to maintain a system
- Maintaining an unsustainable system

Swystems

What is not sustainable?

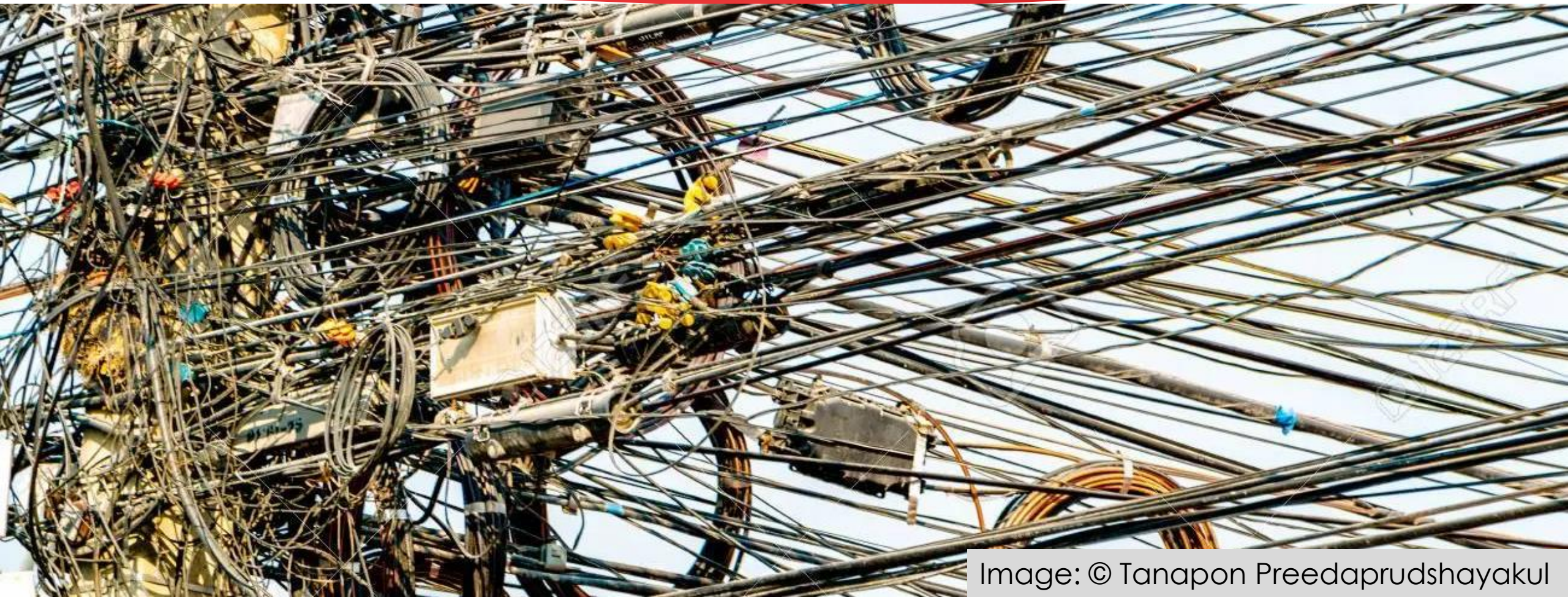


Image: © Tanapon Preedaprudshayakul

Configuration management

Configuration management: establish and maintain consistency between the product and the product requirements and information. [simplified from EIA-649-A]

- Upgrades, reuse, life-cycle extensions and corrective maintenance changes the system's configuration and its requirements
- Tracking changes throughout the system's life-cycle is key to sustain it
- If the system's configuration is not adequately tracked, maintainers will be afraid of changes

- **But:** configuration management may hinder timely and efficient corrective maintenance

Requirements – a formal definition
(Recap from SESE Tour 2021)

- $\mathbb{Y}$ The set of all systems
- $\mathbb{L}_N$ The subset of solutions to a particular specific need
  $\mathbb{L}_N \subset \mathbb{Y}$
- A requirement is a predicate $R_N(y)$ that satisfies

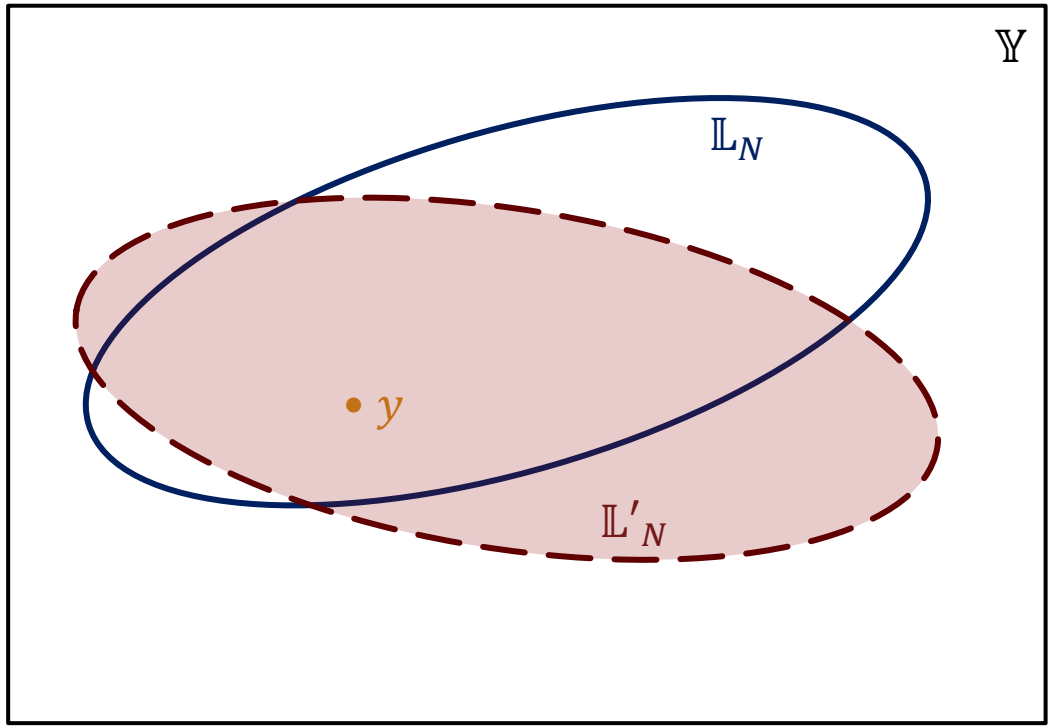
$$R_N(y) \leftarrow y \in \mathbb{L}_N \qquad \forall y \in \mathbb{Y}$$

- A requirement for a particular need is a choice function that holds true for a system that is a solution to that particular need.

Requirements – a formal definition

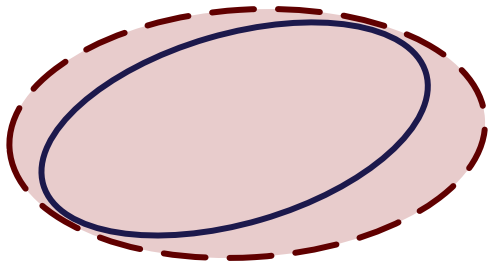
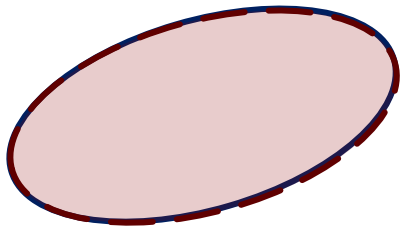
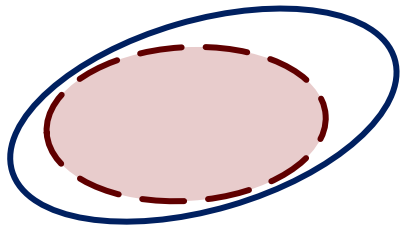
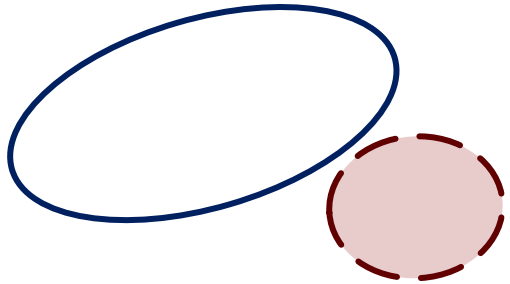
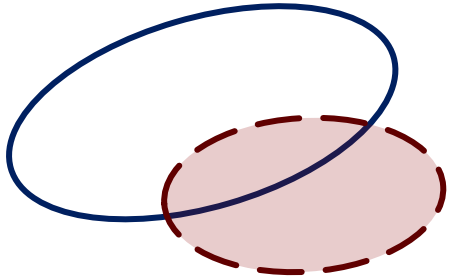
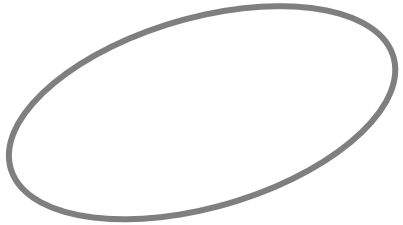
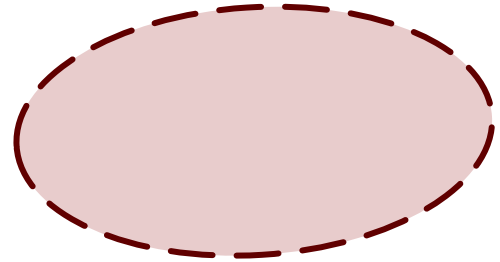
$$R_N(y) \leftarrow y \in \mathbb{L}_N \qquad \forall y \in \mathbb{Y}$$

- $\mathbb{Y}$ The set of all systems
- $\mathbb{L}_N$ Original solution space by R_N
- $y \in \mathbb{Y}$ A particular system
- $\mathbb{L}'_N$ Solution space by modified requirement



Swystems

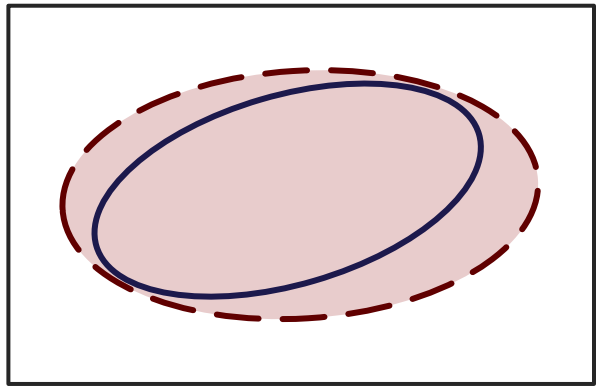
Classification of requirement changes

Relaxed 	Matched 	Restricted 	
Incompatible 	Related 	Removed 	New 

Classification of requirement changes

Relaxed

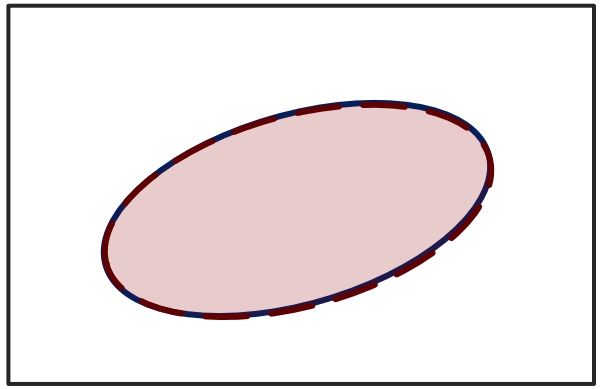
- Requires no change to an already compliant system
- When driven by design or a lower level requirement:
  check that the higher level requirement is still met
- May provide operational simplifications:
  check documentation and operating procedures



Classification of requirement changes

Matched

- Typically due to re-phrasing or requirement clarification usually during design
- In operation, most commonly due to for fit-form-function replacement of a component

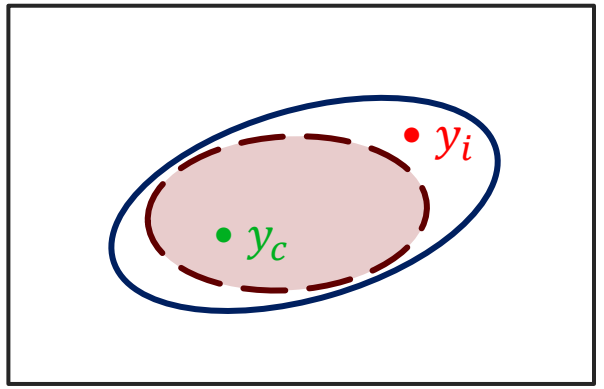


- Requires no change to an already compliant system
- Change in terminology may adapt documentation

Classification of requirement changes

Restricted

- Requires change analysis on existing (or planned) system:
  - *Compatible* restriction: existing system satisfies the restriction
  - *Incompatible* restriction: existing system does not satisfy the restriction

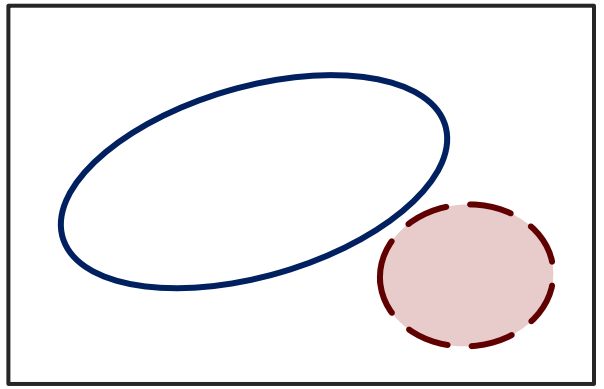


- Compatible restriction: can be handled like a matched change
- Incompatible restriction: may impact design, production, operation, costs …
  → requires full impact analysis

Classification of requirement changes

Incompatible

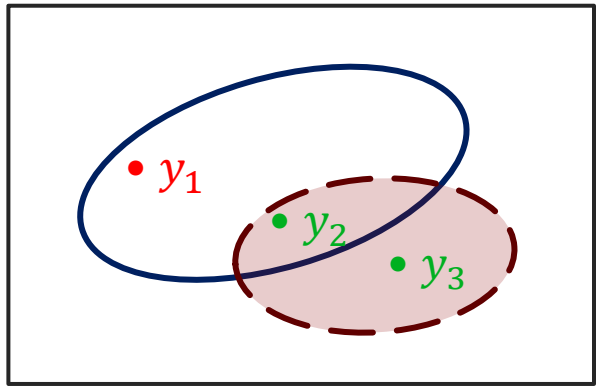
- Any existing solution is no longer a solution to the need
- May be due to a waiver ("retrofitting the requirements"), or an *incompatible* request



- Incompatible requests affect design, production, operation, costs, … and require full impact analysis

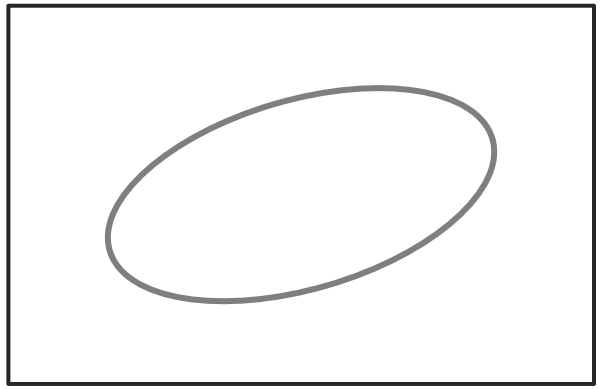
Classification of requirement changes
Related

- Requires change analysis:
  - **Case 1:** Existing (or planned) system is no longer compliant
    → incompatible change case
  - **Case 2:** Existing (or planned) system is compliant to existing and changed requirement
    → compatible restriction case
  - **Case 3:** Existing (or planned) system becomes compliant
    → relaxing change case
- Each case should be adequately addressed



Classification of requirement changes
Removed

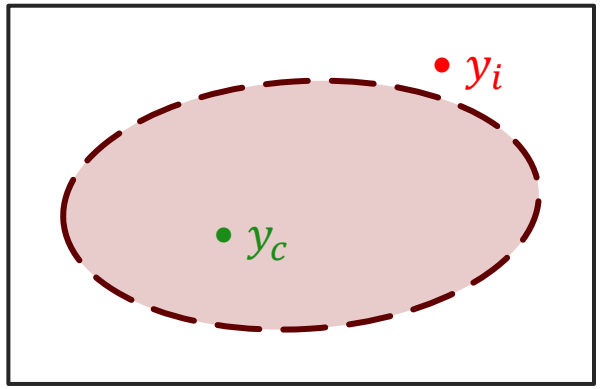
- Basically a relaxing case, where the requirement changes to TRUE



- Requires no change to an already compliant system
- When driven by design or a lower level requirement: check that the higher level requirement is still met
- May provide operational simplifications: check documentation and operating procedures

Classification of requirement changes
New

- Basically a restricting case, where the (previously non-existing) requirement has been TRUE
- Requires change analysis on existing (or planned) system:
  - *Compatible* new: existing system satisfies the requirement
  - *Incompatible* new: existing system does not satisfy the requirement



- Compatible new: can be handled like a matched change
- Incompatible new: may impact design, production, operation, costs …
  → requires full impact analysis

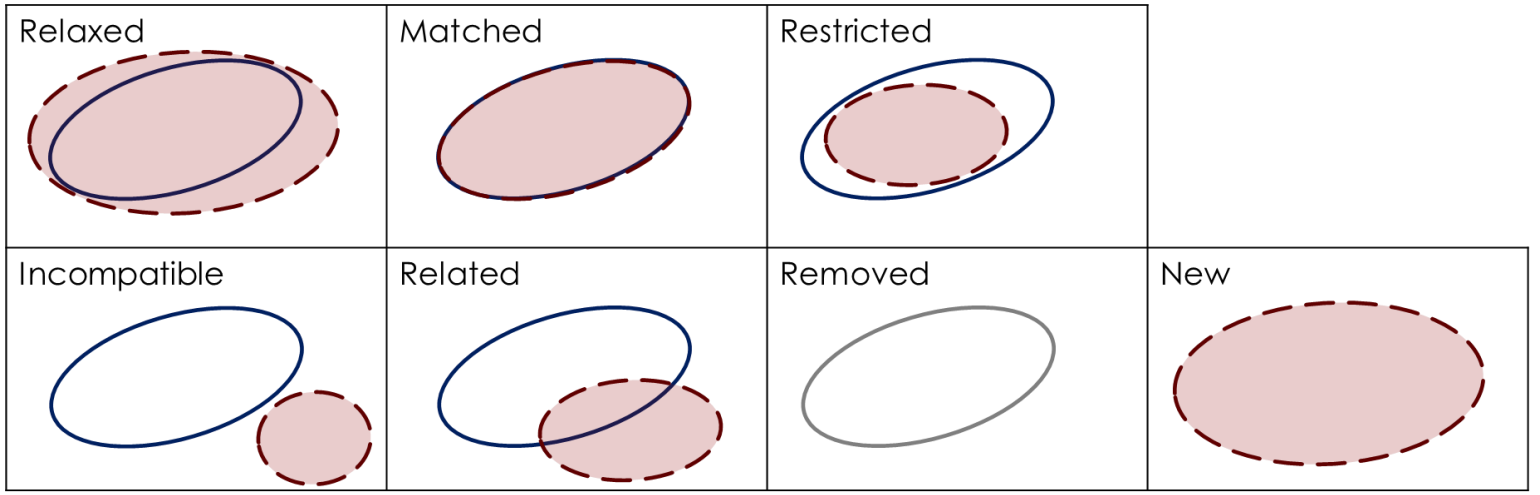
Impact on Configuration Change Management

The presented model

- Is conceptually easy: easy to understand, easy to use, easy to apply
- Can (significantly) speed up configuration changes
  - Many cases require less analysis and effort, freeing capacity to difficult changes
  - Less overhead
- In practice, the classes can be project- and system-specific tailored for even higher efficiency
- Keeps configuration management effective
  - No need to work around configuration management to apply urgent changes
- Contributes to lasting practical sustainability

Conclusion

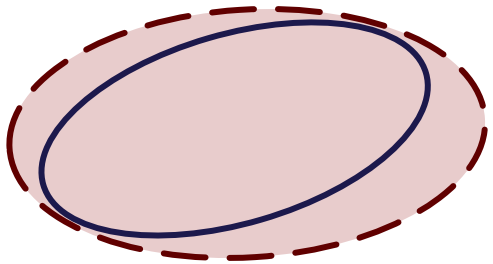
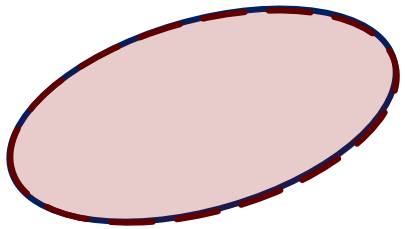
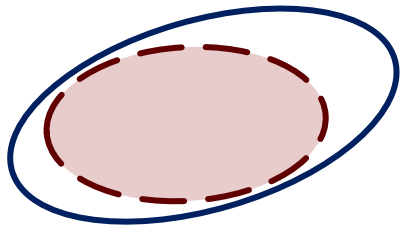
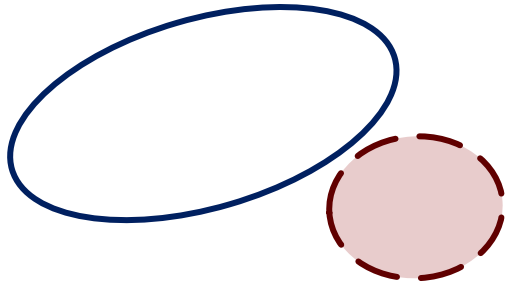
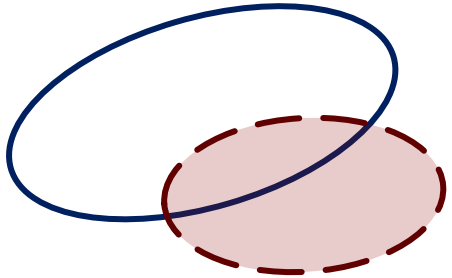
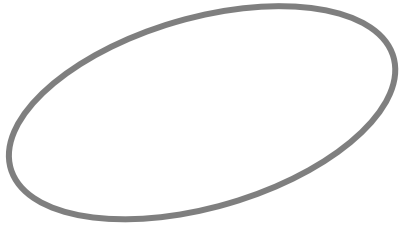
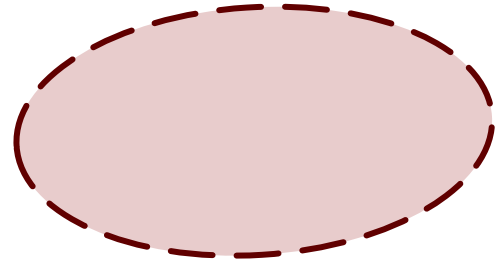
- Use configuration management to keep a sustainable system sustainable
- Not every change has the same impact
  - use requirement changes to classify configuration changes
  - keep configuration change management effective and efficient



Opportunity was designed for 90 days
and sustained for 5‘111 days

Image: © NASA

Discussion

Relaxed 	Matched 	Restricted 	
Incompatible 	Related 	Removed 	New 

Marco Chicherio marco.chicherio@swystems.ch Swystems GmbH, Ankengasse 3, CH-8902 Urdorf