# Industrial challenges

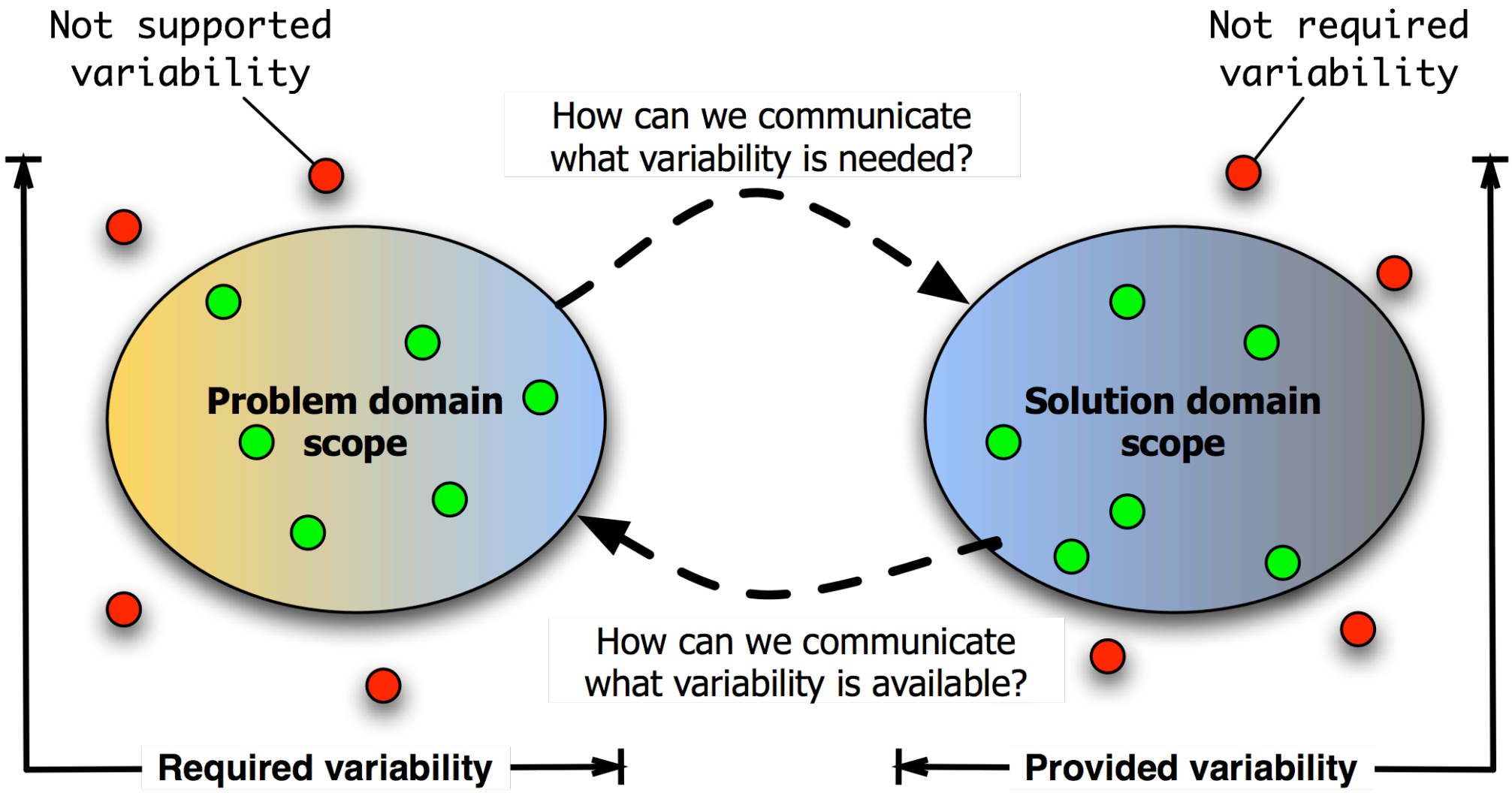
Juha Savolainen

# Outline

- What are product lines and variability management?
- Product lines - required and provided variability
- Products and product lines
- Reuse of requirements
- Conclusions

Please note that all information and examples in this slide set are based on the public domain knowledge of the respective domains. This example is only given for demonstration and should not be used for other purposes. In particular, this presentations is not based on knowledge of future products or their properties. The analysis is made by the authors and does not represent the opinion of our employers.

# Required and provided variability






NOKIA
Connecting People

# Sentences that customers never say...

- “Please maximize software reuse when making the product that I want”
- “Yes, you can discard the feature that I want in order to automatically derive my product”
- “Yes I understand that I have to select that unnecessary feature in order to get the feature that I want”
- “And yes, of course, I will pay for the unnecessary feature”

- **Customers care about products not product lines**
- **It is our job to find the balance between product requirements and reuse**

# Reusing requirements

- What is the ultra reusable requirement?

| ID | Requirement |
|---|---|
| Req.1 | The system shall be good |

- Can be reused as a requirement for nearly all possible systems
  - (Well maybe not Catbert's evil downsizing machine)
- Unfortunately **the most reusable requirements are often the most worthless to reuse**

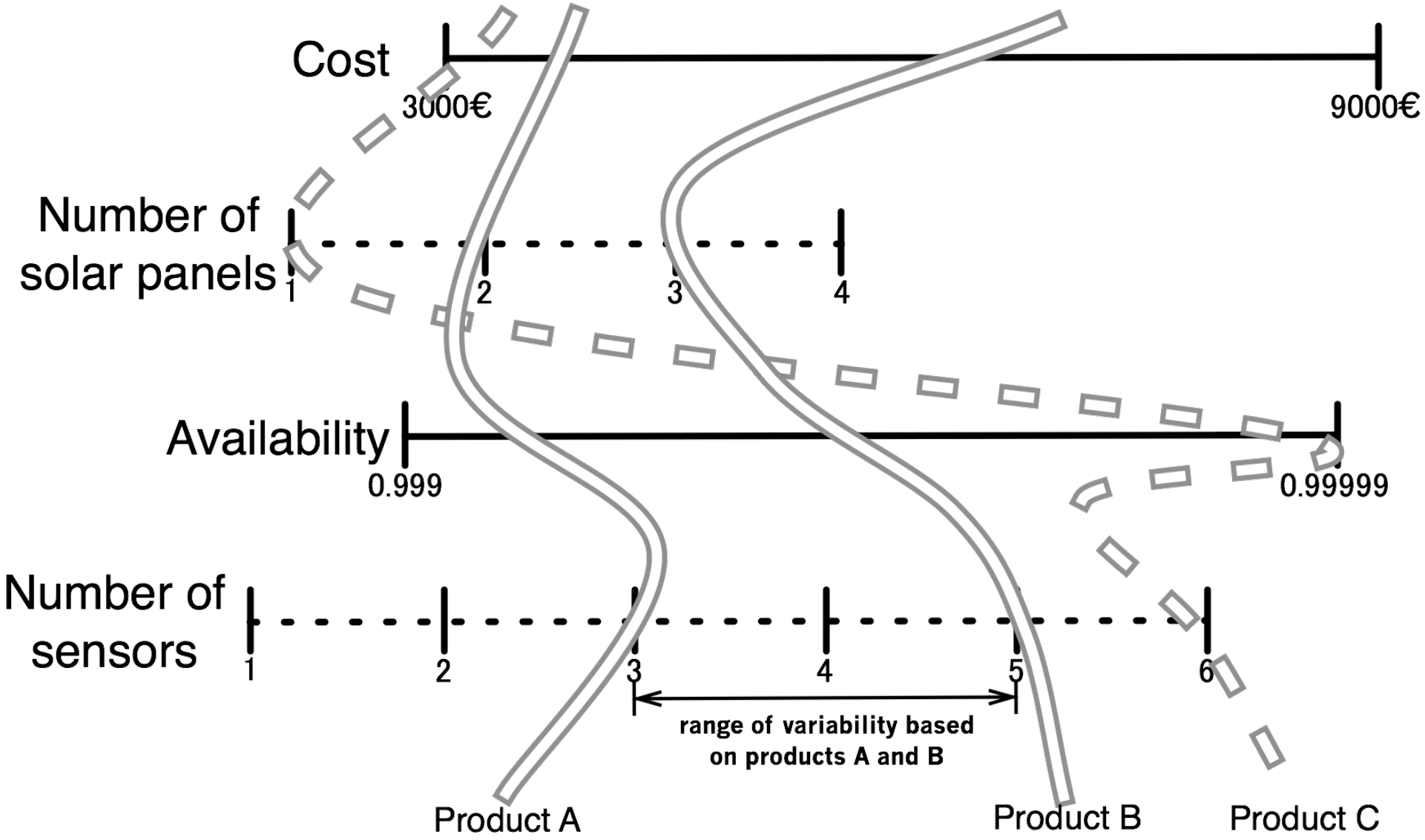
# Requirements reuse

- Okay, to maximize the value of reuse we can then make requirements more concrete

| ID | Requirement |
|---|---|
| Req.2 | The system should present the total amount of US$ on the customers only account in 150 ms in the right hand top corner of the 12” LG1200 ATM screen if the customer has been correctly authenticated using the PIN code that is compared against the verification code gathered from the Nordea bank’s central computer, transmitted over the SSH encrypted channel that has been authenticated with .... |

- This is great value for reuse, if you happen to build ATMs for Nordea bank that happen to be exactly the same as the previous ATMs
- Therefore, **the most valuable requirements that you can reuse and the hardest to reuse**

# Products, variability dimensions, parameters of variation






NOKIA
Connecting People

# Rules to rescue, right?

- Power consumption = 10W * number of wind sensors (nWS) + 12 W * number of cloud height sensor (nCHS) + power of the weather station (pWS)

# Rules to rescue, right?

- Power consumption = 10W * number of wind sensors (nWS) + 12 W * number of cloud height sensor (nCHS) + power of the weather station (pWS)
- Ups, but the pWS must be dependent on the particular HW we are using

# Rules to rescue, right?

- Power consumption = 10W * number of wind sensors (nWS) + 12 W * number of cloud height sensor (nCHS) + power of the weather station (pWS)
- Ups, but the pWS must be dependent on the particular HW we are using
- Ok, so pWS= CPU MHz * the CPU base consumption + amount of memory * memory base consumption

# Rules to rescue, right?

- Power consumption = 10W * number of wind sensors (nWS) + 12 W * number of cloud height sensor (nCHS) + power of the weather station (pWS)
- Ups, but the pWS must be dependent on the particular HW we are using
- Ok, so pWS= CPU MHz * the CPU base consumption + amount of memory * memory base consumption
- But that's not right! The MHz is a variable and it depends on the how much functionality we are running on it

# Rules to rescue, right?

- Power consumption = 10W * number of wind sensors (nWS) + 12 W * number of cloud height sensor (nCHS) + power of the weather station (pWS)
- Ups, but the pWS must be dependent on the particular HW we are using
- Ok, so pWS= CPU MHz * the CPU base consumption + amount of memory * memory base consumption
- But that's not right! The MHz is a variable and it depends on the how much functionality we are running on it
- Ok, so if we annotate all use cases with power usage information, if there is variation then maybe we should use state charts and estimate how may times we do certain activity for a normal day and ... and ...

# Rules to rescue, right?

- Power consumption = 10W * number of wind sensors (nWS) + 12 W * number of cloud height sensor (nCHS) + power of the weather station (pWS)
- Ups, but the pWS must be dependent on the particular HW we are using
- Ok, so pWS= CPU MHz * the CPU base consumption + amount of memory * memory base consumption
- But that's not right! The MHz is a variable and it depends on the how much functionality we are running on it
- Ok, so if we annotate all use cases with power usage information, if there is variation then maybe we should use state charts and estimate how may times we do certain activity for a normal day and ... and ...

- **Okay, maybe modeling everything and all combinations is not such a great idea if you still have to make also products**

# No wait a minute... What were the customer's needs?

- So what the customer really wanted?

- **"A farmer want to know when to put the seeds into the soil"**
- Possible solutions:
  - A personal weather station with customized algorithms to calculate the right movement in time
  - A service provided using a mobile phone (SMS send at the correct time)
  - A personal weather station with internet service that combines data from other regional weather station to provide the service to the farmer
  - ...

- **Sometimes the eagerness to reuse existing assets makes us to ignore new, better ways, to fulfill the customers needs with possible a widely different business model**

# Key industrial challenges and my hunches how to solve them

- How to get great products that differentiate from each other and from the competition
  - Need for a product view that identifies critical features for each product
- How to create those products while maximizing reuse
- How to expand the scope of the product line easily
  - Probably requires combination of centralized integration-oriented approach to handle required variability and composition-oriented approach to handle provided variability
- Understanding what to model and what not to model
  - Complete automation and the required effort has rarely enough ROI to be considered
  - Model only critical parts and rely on the intelligence of developers and customers for others
- No hugely complex king models
  - Rather a number of interconnected models that concentrate on one view of variability
  - However having many models with complex dependencies may be even worse...