

Requirements in Software Product Lines

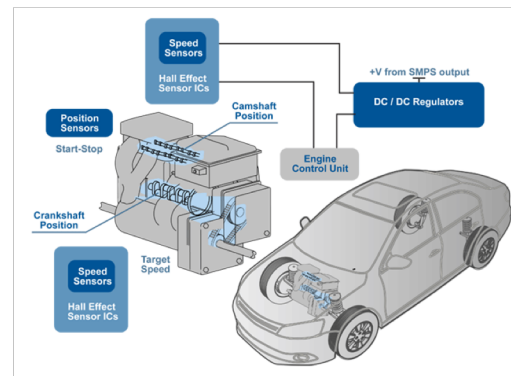
Ivo Indergand und Laura Salathe

Group C
Requirements Engineering II

What is a software product line?

“A software product line is a set of software-reliant systems that share a common, managed set of features satisfying a particular market or mission area, and are built from a common set of core assets in a prescribed way.”

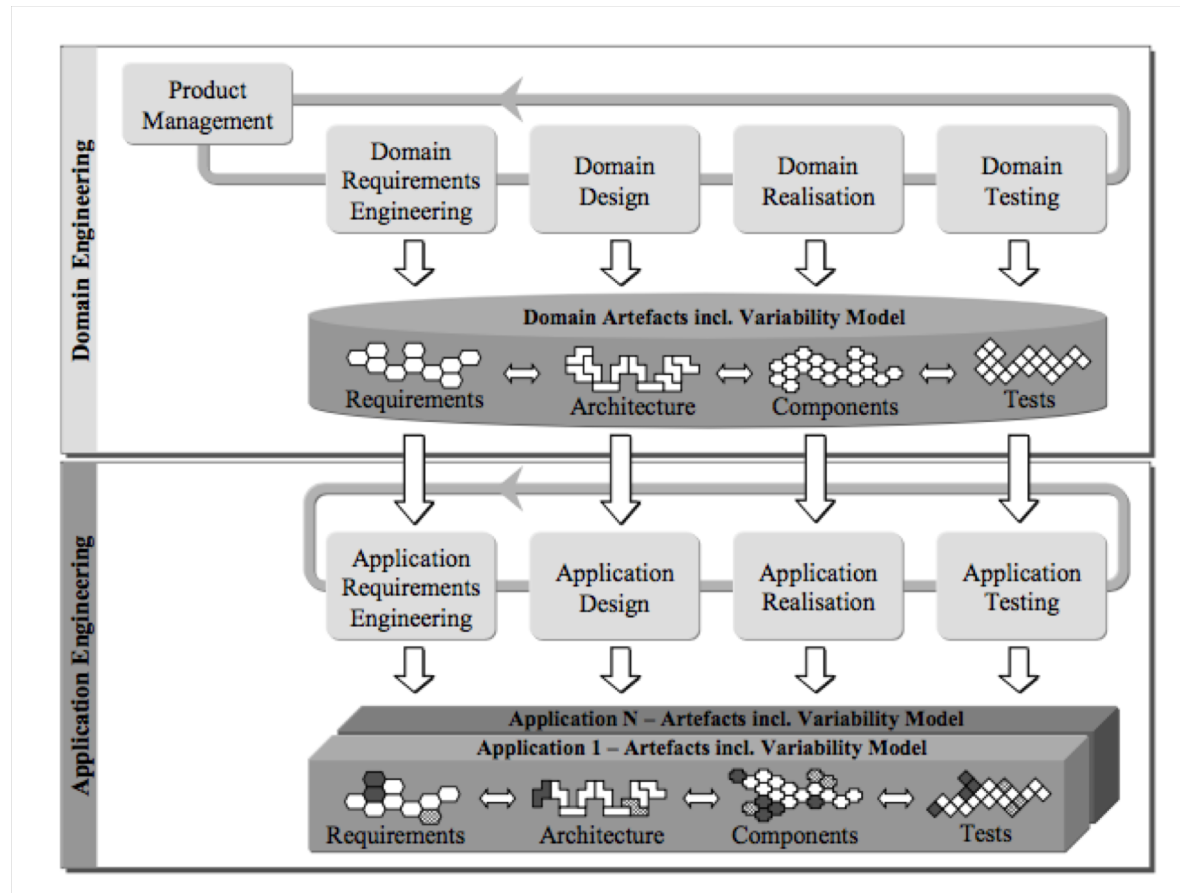
Software Engineering Institute - Carnegie Mellon University



What about models?

- Suitable models are always very important for a good result
- There are specialized models and frameworks for product line engineering
- This models are part of this presentation

What is special about product line engineering?



Böckle, Pohl and von der Linden (2005)

Main goals

Domain Engineering:

- Define the commonality and the variability of the software product line.
- Define the set of applications the software product line is planned for, i.e. define the scope of the software product line.
- Define and construct reusable artefacts that accomplish the desired variability

Application Engineering

- Reuse as many domain artefacts as possible
- Document the application artefacts, i.e. application requirements, architecture, components, and tests, and relate them to the domain artefacts.
- Find and document the shared requirements as well as the specific application specification.

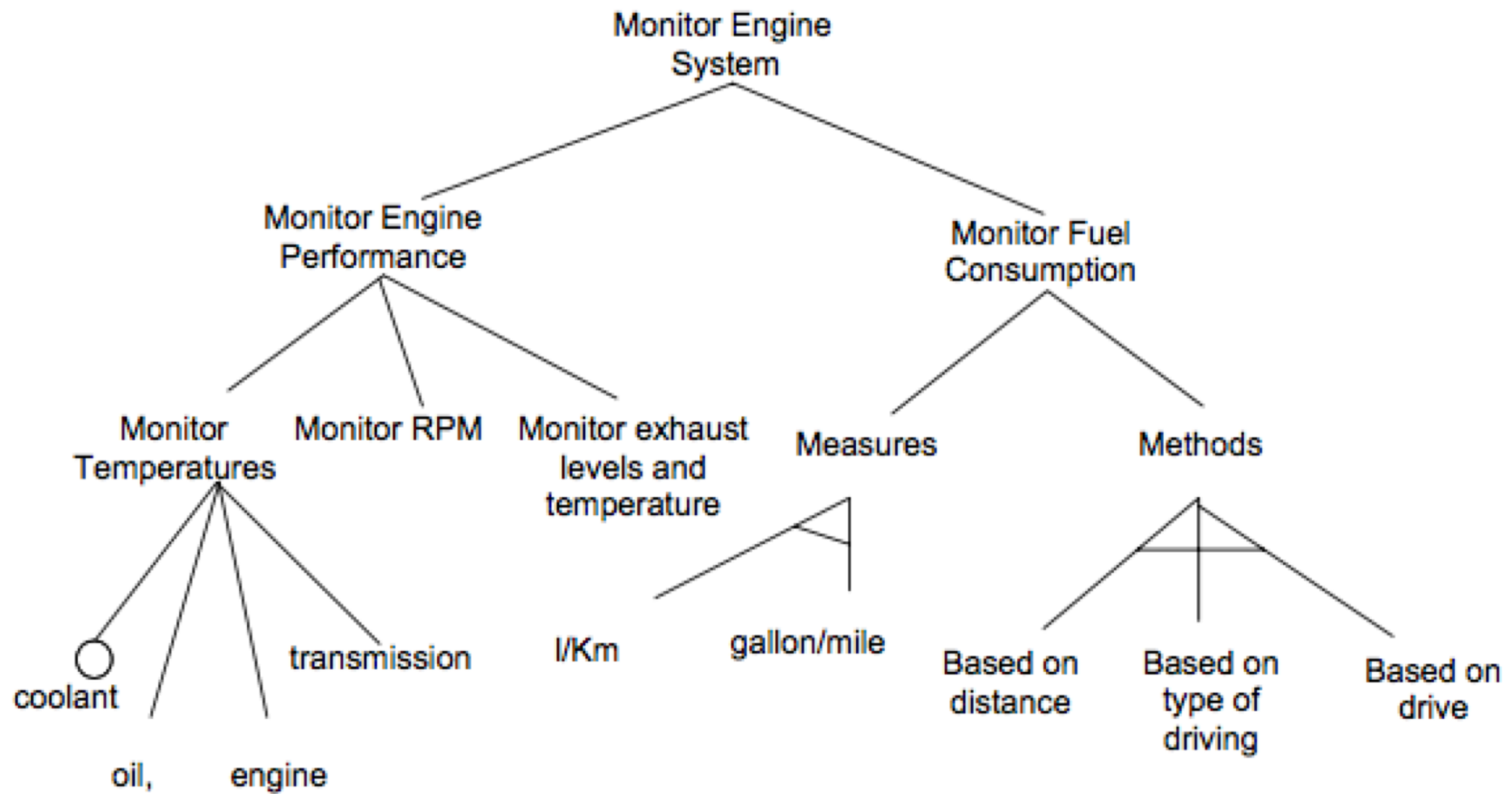
Feature diagrams

- used to express knowledge about allowed feature combinations
- represent feature dependencies
- Evolving features are applied in domain engineering
- Features deployed in many contexts are used in application engineering
- Valuable tool to address the complexity of engineering software product lines
- Less foundations to avoid ambiguities and build efficient tools

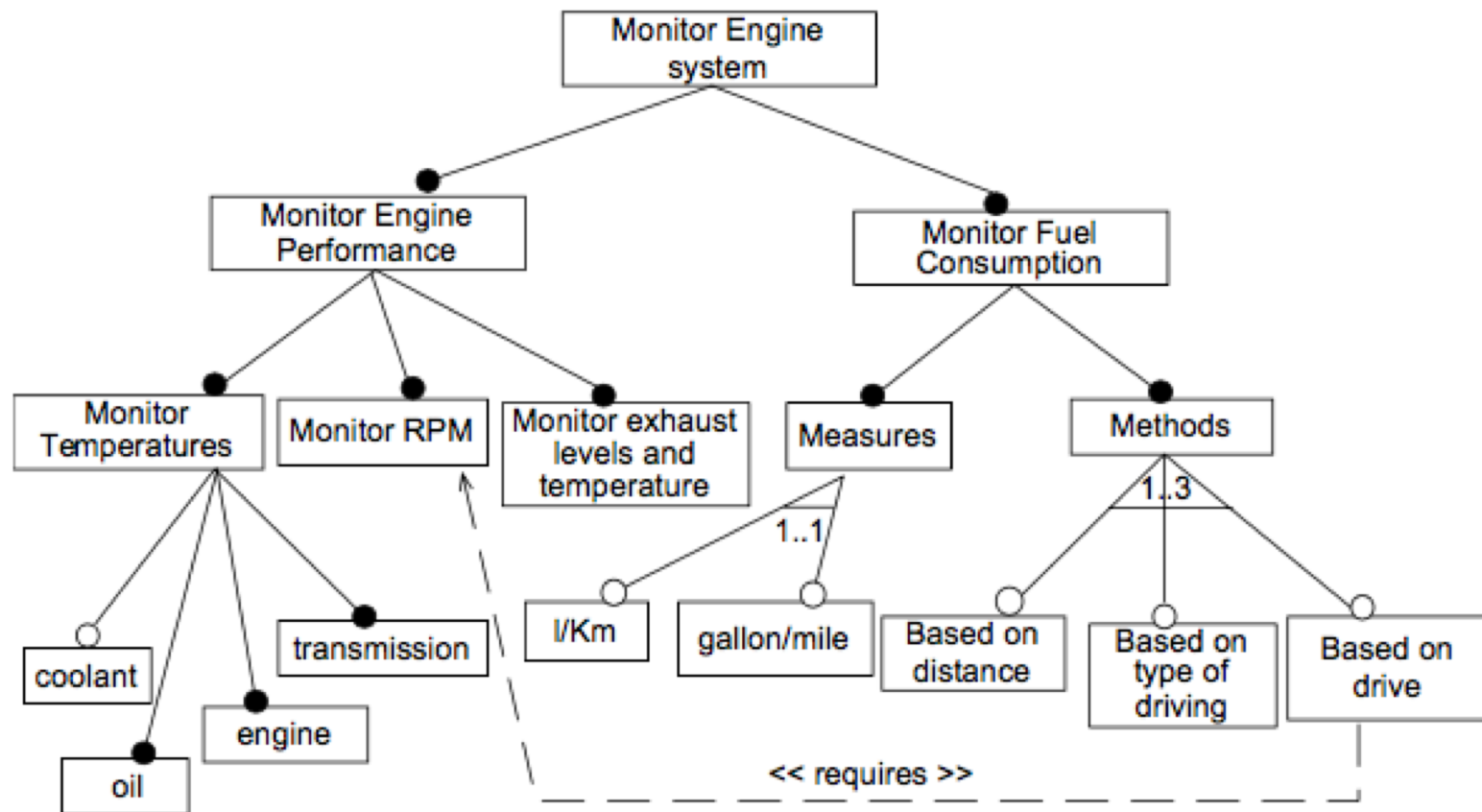
How does a feature model describe PL's variability?

- It is important that rigorous criteria are used to choose the new FD language and that proved correct semantic-preserving translations are used. Several product lines should work with the same modelling languages to avoid additional efforts.
- In the context of emergent and complex systems, it is important that automation of product line and model-driven engineering activities such as those that we just described are made as available, as safe and as efficient as possible.

FODA (OFT) Example



Extended Feature Diagram (EFD)

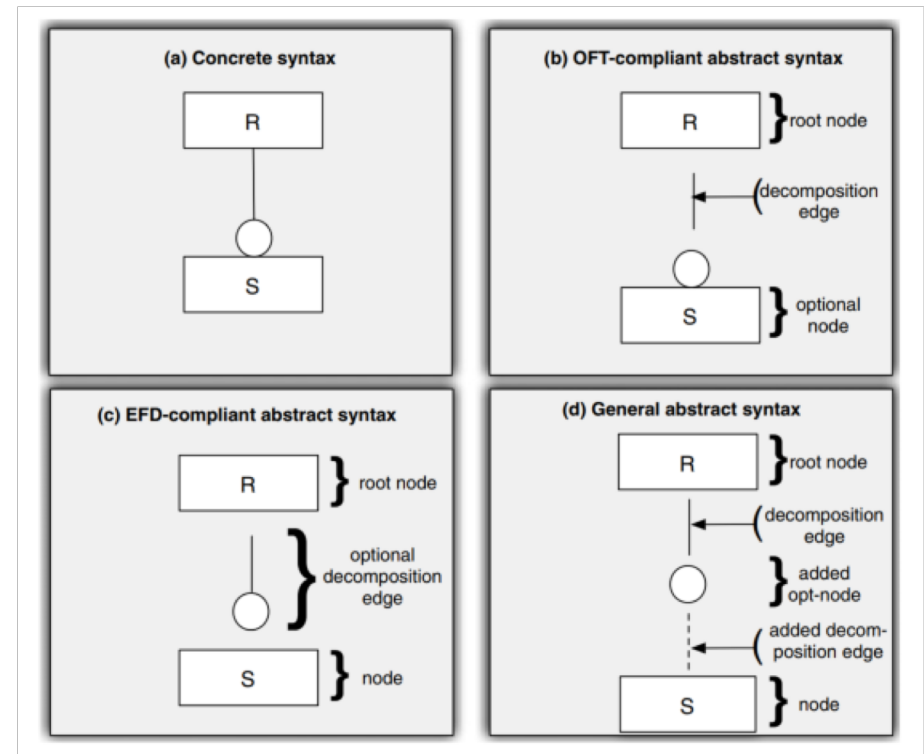


Syntax and formal semantics

- Comparing languages means formalization of syntax and semantics of feature diagram to guarantee that PL and FD are reversely expressible.
- Basis criteria: be expressively complete and closed under Boolean operations. To choose among those, embeddings (avoid redundancy) and succinctness (cardinalities) are inspected.
- Formal comparisons are done by expressiveness, embeddability, succinctness and redundancy.

Generic abstract syntax

- Free Feature Diagrams (FFD): parametric construction generalizing syntaxes of all FD variants
- FD's varying in form (tree/DAG), in allowing operators on nodes or graphical- or textual constraint types
- By providing values for 4 different parameters semantics only have to be given once
- Example: Graph type (GT) parameter is Boolean where 0 is a DAG and 1 a tree



References

K. Pohl, G. Böckle, F. van der Linden (2005). Software Product Line Engineering – Foundations, Principles, and Techniques. Springer. 20-38.

P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, Y. Bontemps (2007). Generic Semantics of Feature Diagrams. Computer Networks 51(2): 456-479.