SPLE: develop and maintain a (software) product line or family from a **shared architecture** or platform and **mass customisation** (e.g. to serve different markets)

⇒ maximise commonalities whilst minimising cost of variations (i.e. of individual products)
Software Product Line Engineering
or Product Family Engineering

SPLE: develop and maintain a (software) product line or family from a shared architecture or platform and mass customisation (e.g. to serve different markets)

⇒ maximise commonalities whilst minimising cost of variations (i.e. of individual products)

Product: a valid combination (configuration) of features

Product line: set of valid feature combinations of a domain
Variability in terms of features:

- stakeholder visible pieces of functionality of a system . . .
- . . .which may be optional and/or may have alternatives
- only specific feature combinations lead to valid products
Variability analysis
Lego examples by S. Apel (U Passau, Germany)

Variability in terms of features:
- stakeholder visible pieces of functionality of a system . . .
- . . . which may be optional and/or may have alternatives
- only specific feature combinations lead to valid products

Feature model: compact representation of all products of a family
Configure your 11-inch MacBook Air

Hardware
Service and Support
Accessories
Printers

Processor
Enjoy incredible performance from fourth-generation Intel Core processors. Choose the speed and processor you want.
Learn more

- 1.3GHz Dual-Core Intel Core i5, Turbo Boost up to 2.6GHz
- 1.7GHz Dual-Core Intel Core i7, Turbo Boost up to 3.3GHz [+ £130.00]

Memory
More memory (RAM) increases overall performance and enables your computer to run more applications at the same time.
Learn more

- 4GB 1600MHz LPDDR3 SDRAM
- 8GB 1600MHz LPDDR3 SDRAM [+ £80.00]

Storage
Your MacBook Air comes as standard with flash storage. Flash storage has no moving parts and provides faster responsiveness and enhanced durability.
Learn more

- 256GB Flash Storage
- 512GB Flash Storage [+ £240.00]
As for CAS, scalability is an issue

Example by C. Kästner (CMU, Pittsburgh, USA)

33 optional, independent features

a unique product for every person on this planet
Deciding a bike-sharing system (BSS) for a city poses many questions

- How many/what kind of bikes (features like light, basket, engine)?
- How many/what kind of stations (capacity), where to place them?
- How to avoid stations being completely full or empty for periods?
- Which BSS features (like maintenance, antitheft, smart services)?
- With or without dynamic redistribution of bikes? And what kind?
- Incentives (rewards) for users bringing bikes to less popular stations?
- Subscription costs and charging policy (like credit card or keycard)?

How to evaluate the numerous options, costs/benefits, improvements and changes in a systematic way? (i.e. behaviour, performance)
Quantitative constraints and attributes

er Beek, Fantechi, Gnesi, Legay, Lluch Lafuente, Vandin @ SPLC/Very*/SEFM

\[
\sum_{\text{feature} \in \text{Features(product)}} \text{price(feature)} \leq 600
\]
Multi-objective optimisation with ClaferMOO: compare system (re)configurations or variants w.r.t. various quality dimensions, e.g. minimising cost whilst maximising customer satisfaction and capacity:
Multi-objective optimisation with ClaferMOO: compare system (re)configurations or variants w.r.t. various quality dimensions, e.g. minimising cost whilst maximising customer satisfaction and capacity:
Multi-objective optimisation with **ClaferMOO**: compare system (re)configurations or variants w.r.t. various quality dimensions, e.g. minimising cost whilst maximising customer satisfaction and capacity:

Address also **behavioural** variability: lift success stories from single product/system engineering to SPLE (e.g. scalable verification like mean field/family-based/statistical analysis techniques)
Product-based analysis

\( O(2^n) \) for \( n \) features
Product-based analysis

$O(2^n)$ for $n$ features

Simple approach
Standard tools available
Infeasible for large product sets
Family-based analysis
Family-based analysis

Beneficial for many products with substantial similarities
Generates complex analysis tasks
Requires (compact) family metamodels
Relating local and global system views

Modal Transition Systems with variability constraints

Performance-annotated UML Activity Diagrams with $\delta$-modelling
Relating local and global system views

Modal Transition Systems with variability constraints

Performance-annotated UML Activity Diagrams with $\delta$-modelling
Relating local and global system views

Modal Transition Systems with variability constraints

Performance-annotated UML Activity Diagrams with $\delta$-modelling

Symbolic computation PAAD up to 3 orders of magnitude faster

$$\text{MTS} \models \phi \Rightarrow \forall \text{LTS} \models \phi$$
FLan, PFLan, QFLan

- process-algebraic operators: specify both configuration and behaviour
- constraint store: specify all variability constraints from feature models
FLan, PFLan, QFLan

- process-algebraic operators: specify both configuration and behaviour
- constraint store: specify all variability constraints from feature models

Statistical model checking:
installation probabilities

⇒ measure likelihood of configurations and behaviour
(e.g. QoS, reliability, adaptivity)
Novelty and roadmap

- Family-based analysis of PAADs: novel application of \( \delta \)-modelling, first attempt to efficient performance modelling of SPLs
- QFLan offers advanced quantitative constraint modelling options: implementation first application of SMC in SPLE
- Preservation results of properties from family (macro) to product (micro) behaviour permits efficient (global) family-based analyses of MTSs instead of costly (local) product-based analyses of LTSs

- Family-based analysis of more complex PAADs with \( \delta \)-modelling
- QFLan: move from the current prototypical implementation to a more efficient, scalable, and robust tool
- Quantitative evaluations of scalability of verification approaches