Matching your Software Variability Challenge

This opportunity scout identifies the opportunity you want to address and check whether configuration language can be of assistance.

Analysis of your business challenges

In the business analysis we select the business challenges emanating from software variability that can be tackled using a configuration language approach. A couple of well known business challenges are described as an example.

- **cost reduction**: *"How to avoid expensive software customization for creating software variants?"*
  => In traditional SE customization is expensive as it requires in-depth knowledge of the software. CL abstract from these technical difficulties while keeping the configuration sufficiently more expressive compared to plain configuration.

- **better products:** *"How to enrich your product by spending more time in product development and less time in product tweaking?"*
  => In traditional SE customization takes a lot of time as it requires in-depth knowledge of base product. CL's abstractions outperform the insufficient abstractions of general-purpose languages and offer a rich set of application domain concepts to work with.

- **faster delivery:** *"How to increase your time to market?"*
  => Compared to traditional SE there is no traditional software development cycle, no testing cycle as there no bugs or features that can interfere with your application.

- **growth, more customers:** *"How to serve more customers with less product development?"*
  => Compared to traditional SE it is impossible to maintain 1000 products, but it is feasible to maintain 1000 configurations.

- **improved market position, stronger customer intimacy:** *"How to better serve your customers and outsmart your competitors?"*
  => With traditional SE competitors will struggle to keep up with the variants among the demands of their customers.

- **increase your market share, distribution channels:** *"How to sell your code base and out-source customization in the form of configuration?"*
  => Traditional SE creates a customer-vendor-lockup which can only be unlocked by a difficult and thorough process of compliant & controlled modification.

- **risk reduction:** *"How to reduce the risks of software customization when facing open-ended requirements?"*
  => Traditional SE does not preserve the requirements of software, they are encoded into general-purpose language abstractions. Therefore one heavily relies on testing before any product or application can be shipped. CL are tested upfront, does not change the code base in expectant ways. CLs are safer to use. The explicit configurations of CLs also supports rigorous understanding of core domain concepts and activities throughout the development chain (customers, requirements engineer, developer, ..) by improving mutual understanding, avoiding incorrect interpretations.

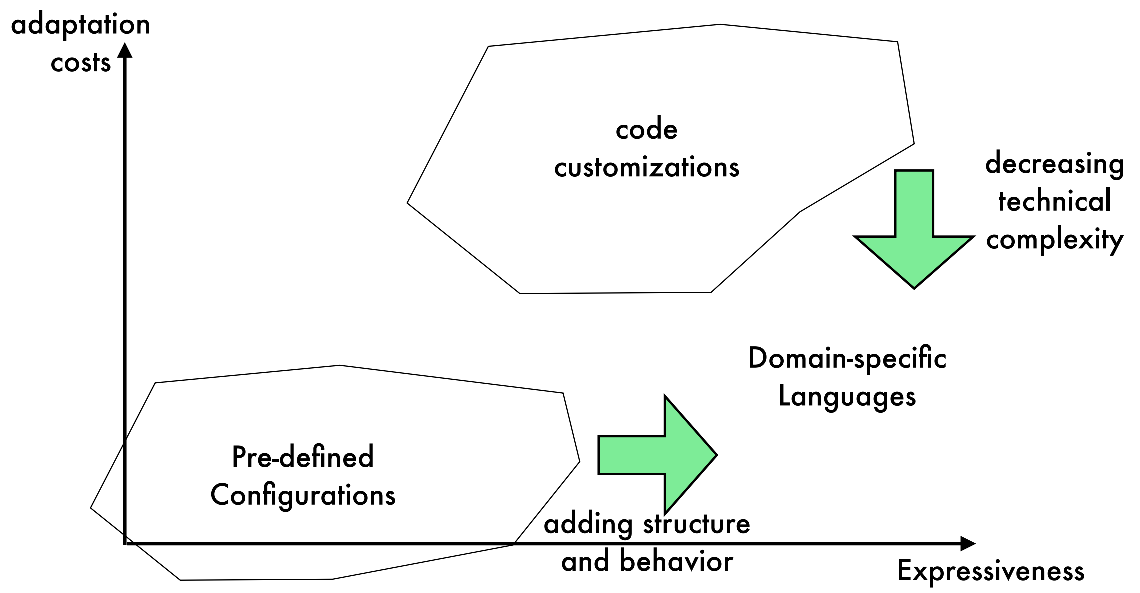
Technical analysis of the software variability challenges faced by a company:

In the technical analysis we select the software variability challenges that can be tackled using configuration language expertise. The items printed in italic and aligned on the right are the materials we require to perform the technical analysis more in depth.

We investigate the position of three dominate technologies for addressing your configuration challenges:

1. Pre-defined configurations
2. Code customizations using traditional software engineering
3. Advanced configurations via domain-specific languages

We do this by determining the positions of the first two technologies, and subsequently investigating whether the forces affecting your product development cause you to consider advanced configurations via domain-specific languages.


adaptation costs
code customizations
decreasing technical complexity
Domain-specific Languages
Pre-defined Configurations
adding structure and behavior
Expressiveness

We distinguish between two kinds of stereotypical software businesses: product-based and project based. In product-based business aims build a (single) software product that can be used by different clients. The focus during development is to construct a core of shared functionality among clients while handling requests for customer specific functionality. In project-based business the focus during development is to construct a separate product for each client.

For each kind of stereotypical business we formulate important considerations in form of questions, answers and the required materials necessary to properly analyze your variability challenges.

- Product-based business
  - In the family of products:
    - How does each product variant differ from another one. How do they differ both in structure and behavior?

      Product Family Description
      A couple of Product Descriptions for each Product Variant
      Examples of Product Variations

  - Who is/should be creating product variants (Developers / end-users / other customers) is struggling with:
    - Product variants are not created easily, or
    - Product variants are not created correctly.

User Experiences
Existing product variations

- When product variants are created:
    - A formal client request enters but is there sufficient time to create the variant?
    - The product is running and should be created on the fly?

Client requests

- The method to support variability is limited because:
    - Classical parameterization (choices, parameters, xml files, …)
        - Are not sufficiently rich/expressive ?
        - Are not easy to understand or to maintain ?
    - The software engineering method used for supporting variability (copy-paste, assembling modules, customizing modules)
        - Is using a common code base. However what is the degree of sharing between individual products, what is the degree of modularity?
        - Is using the best technology so far to avoid code duplication. However, what is the degree of repetitiveness that still remains?
        - Is using a standard process for creating a product variant. However, what is the degree of maturity? Degree of consistency?

Product variant creations
Product code base
Product configurations

- Project-based businesses
    - There is a common code base. What is the degree of sharing between individual products? Degree of modularity?

Product variant creations
Product code base
Product configurations

Operational analysis of the software variability challenges faced by a company

- What is fixed in your working and operating environment (tools, methodologies, software, legacy code, devices, ...)?
- What resources (time, money, personnel) are available to invest in variability?

Tools & Programming languages
Methodology
Development Cycle
Personal availability

Offer

- Help you to understand how you can manage your configuration to meet your business goals and that is appropriate for your product assets.
- Listing the necessary steps (starting from the prerequisites up to a working prototype) on how to tackle your challenges using configuration languages.

Contact:

Dr.-Ing. Sebastian Günther
sebastian.guenther@vub.ac.be

Dr. Thomas Cleenewerck
tcleenewerck@gmail.com

Vrije Universiteit Brussel I Department of Computer Science I SOFT Research Group

Pleinlaan 2 I 1050 Brussels I Belgium