A Roadmap To Variability-Aware Design of Database Technology

Phd Student
Selma BOUARAR
Selma.bouarar@ensma.fr

Advisors
Ladjel BELLATRECHE
Bellatreche@ensma.fr
Stéphane JEAN
jean@ensma.fr

Team Seminar
06/11/2014
PLAN

- Context:
  - Software reuse
  - Database design

- Objective
  - Database as a Software Product Line

- Contributions
  - A SPL-based framework for designing DB
  - Evaluation of the proposal

- Future work
Most organizations produce families of similar systems, differentiated by features.

A strategic reuse makes sense.

Reuse History

- Focus on Mass customization and domain knowledge
- Domain-driven approaches: Generative programming, Software Factories and Software Product Lines (SPL)
“SPLE is a paradigm to develop software applications using a set of common assets and mass customization” [Pohl et al. 05]

- From Ad hoc to Systematic reuse

- Widespread Use of Software Product Lines
The data flowing through all the organizations is managed by database systems. E.g. Static Data, scientific, aggregated, etc.

Context: Software engineering vs Database systems

- Requirements
  - Conceptual Design
  - Logical Design
  - Physical Design

- Storage/Management
  - Statistic DB
  - Scientific DB
  - DW

- Deployment
  - Centralized
  - Flash
  - Cloud
  - Tuning

Development cycle (Software)

Variability-Aware design of Database Technology
Motivation and objectives

- Variety of choices
  - SPL to manage variability all along the cycle
  - Enjoy benefits: Reduced time to market/cost, Quality, maintainability, etc.
- From an ad hoc DB design to a systematic one
- Deal with database design as a whole
- Assist DB users during design process
Issues and Contributions

Question 1: How to adapt SPL for Database design

Answer: Identification of the artefacts

Issue: An evolving lifecycle

- A generic framework
- Formal definition of database lifecycle
- Dependencies between variants

Contribution 1: A SPL-based framework for designing highly customizable database products
Issues and Contributions:

1-A SPL-based framework for designing DB

- Identification of the artefacts making up a SPL

- Market Strategy / Application Domain pertain to

- Objective

- Contributions

- Future work

Any organisation having data to store/manage is satisfied by

- Precedence of phases

- Architecture

- Constraints

used to structure

- Variability of the design life cycle

Products

Concrete instance of a DB ready-to-be-implemented

Modern Strategy / Application Domain

Components
Issues and Contributions:

1-A SPL-based framework for designing DB

- An in-depth study of the life cycle
  - Generic Formal definition
- Moving from formal definition to feature models
- Variability across DB design life-cycle
Issues and Contributions:

1-A SPL-based framework for designing DB

- Constraints, dependencies
  - Reason on feature models
  - Decrease possible configurations number
  - When to bind to a variant (Compilation, Runtime, etc.)

- Specification of dependencies between variants
  - Deployment Architecture requires semantic
  - Logical design requires a conceptual design
Issues and Contributions:

1- A SPL-based framework for designing DB

- **Implement features**
  - A set of Java classes

- **Configuration**
  - Binding variation points to variants
  - Feature-oriented programming with Jak files

- **Eclipse Plugin**
  - FeatureIDE [1,2]

- **Maintenance**
  - Populate variant features
  - Prune old variants
  - Adapt to new requirements
Question 2: How to assist DB designer, during the configuration process, against the panoply of choices?

Answer: Evaluate variants and compare

A specific type of testing

Issue: Large number of variants

Contribution 2: Evaluation of the proposal: Use case on testing DB design
Issues and Contributions:
2- Use case on testing databases

[Diagram with nodes and edges representing concepts related to database technology and testing]

- Testing concern only leaf features

- Configuration
  - Design
  - Feature
  - Process

- Performance
- Functionality
- Usability

- Context
- Objective
- Contributions
- Future work
Issues and Contributions:  
2- Use case on testing databases

Logical Design:
- Embedded Variants
- First Level: Tied to DBMS choice.
- Variation point: Relational
- Variants: NF1, NF2, ..., BCNF
- Input: $CM_i = \varphi(CM, F_i)$
- Output: $LM_1, LM_2, ..., LM_n$ / Variability implementation
- Test:
  - Performance, Functional, Usability, etc.
- Result: Choose the most suitable $LM_i$
Issues and Contributions:

2- Use case on testing databases

✓ Input: \((F, D_1, \ldots, D_n) + \{Q_1, \ldots, Q_n\}\).

✓ Dimension having \(h\) hierarchical levels can be decomposed \(2^{h-1}\) times
  
  ▪ E.g. \(\text{Dim (location)}=\{\text{Region, Nation, city}\} \Rightarrow 2^{3-1} = 4\)

\(\prod_{d=1}^{n} 2^{h_d-1}\) possible schemas
  
  ▪ For 4 dimensions, each having 3 hierarchical levels \(\Rightarrow 256\) schémas possibles

\(\Rightarrow\) Necessity of a cost model to evaluate these schemas
Issues and Contributions:

2- Use case on testing databases

- **Deployment Layout** [Jean et al., 13]
- **Optimization structures:**
  - [Boukorça et al., 13]
  - Choice of the algorithm
Question 3: From Ad hoc to systematic design, how can this help in the predictability of DB schema evolution

Future work

Contribution 3: Application of SPL-based database design on Schema evolution
References


MERCI

Des questions... ?