Using Normalized Systems Expansion to Facilitate Software Migration - a Use Case

Christophe De Clercq
Jan Verelst

fulcra
Transformation Architects

Naam:

- Christophe De Clercq

Functie:

- Architect and Managing Partner

E-mail:

- christophe.de.clercq@fulcra.be

Telefoon:

IARIA

On Software Migration - Why

- **When technology on which the system is developed is no longer supported**
- **When it becomes too pricey and resource-consuming to support**
- **When the system does not correspond to the renovated business processes**
- **When the system cannot be integrated with modern tools**

Laws of software evolution of Manny Lehman

Continuous Change

Increasing Complexity

Growth in Features

Declining Quality

Hidden Feedback

On Software Migration - approaches

Database First

↓

App migrated incrementally

Database Last

↓

App migrated first

Composite database

↓

Both directions

Chicken Little Strategy

↓

Composite functionality

Big Bang Methodology

↓

From Scratch

Butterfly Methodology

↓

Mission critical

Using Normalized Systems Expansion to Facilitate Software Migration - a Use Case

On Normalized Systems

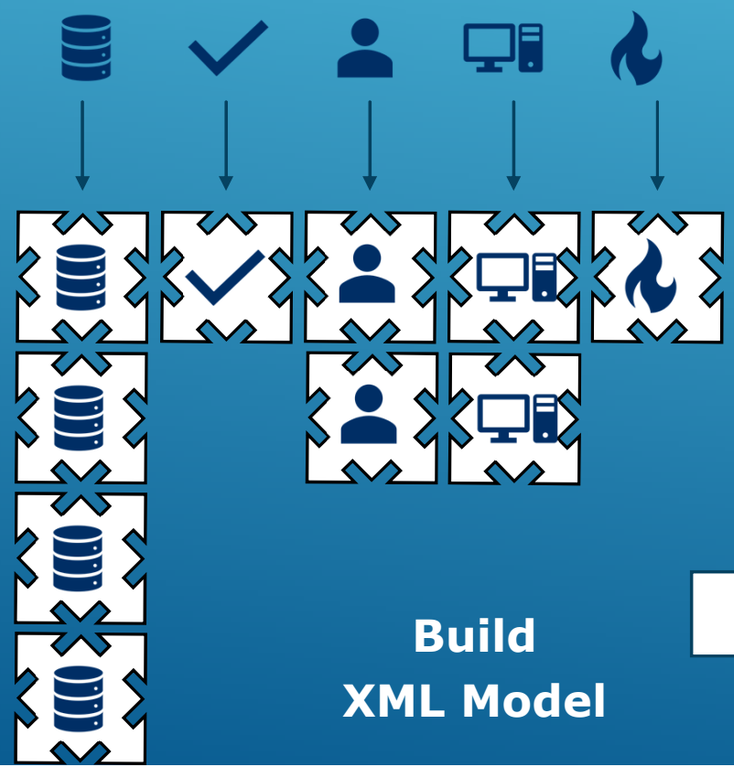
Separation of Concerns Change driver separated from other concerns

Data Version Transparency without impacting the input or output

Action Version Transparency Without impacting the calling components

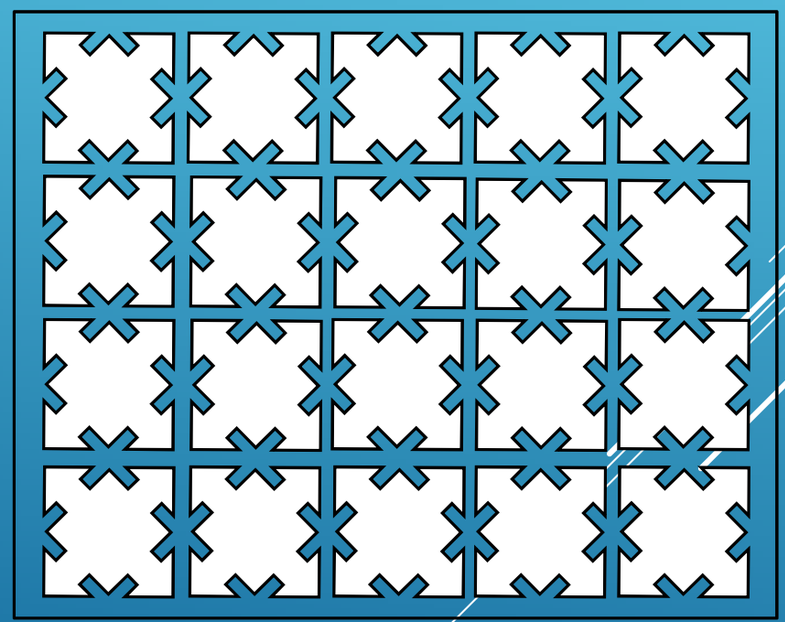
Separation of States each step separated in time

Requirements are converted into n instances of each element



Build XML Model

Expand



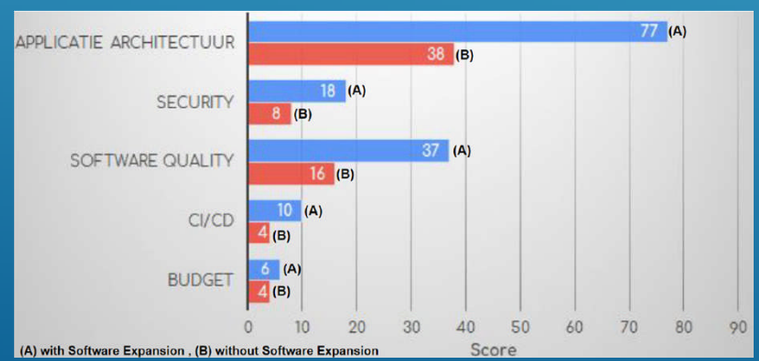
NS application = n instances of elements

Use Case: Connecting Expertise - context

Connecting Expertise offers a job matching platform (called CE)
Used by companies who need resources (job seekers) and those who offer resources (job suppliers)

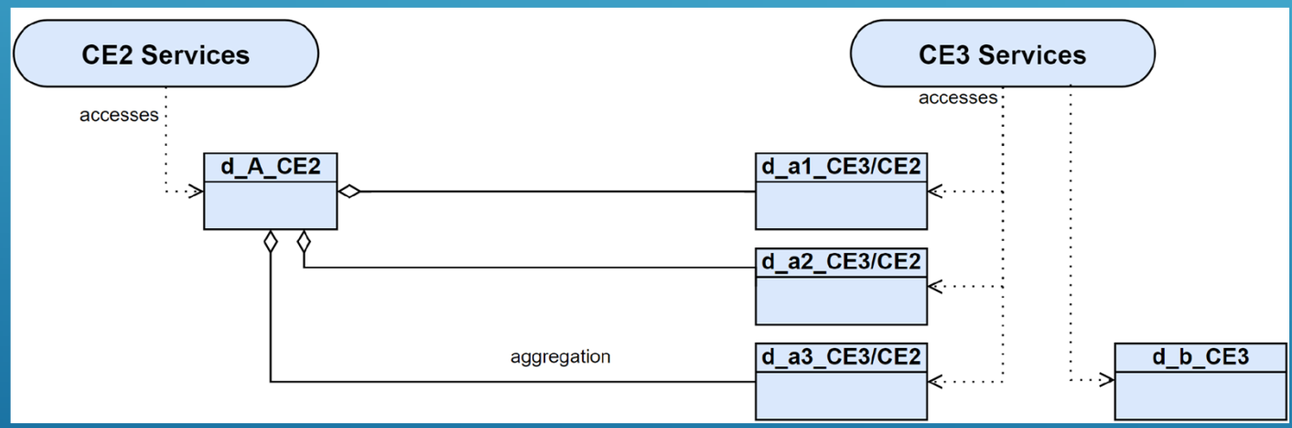
CE required a major upgrade because:
- New user cases: API based integration with systems of the job seekers and suppliers

But facing increasing difficulties in extending their current platform because:
- Issue with code quality (lack of standards, duplication, tight coupling, lack of tests etc.)
- Issue with code complexity making end-to-end testing complex
- Issues with naming conventions
- Issues with scalability



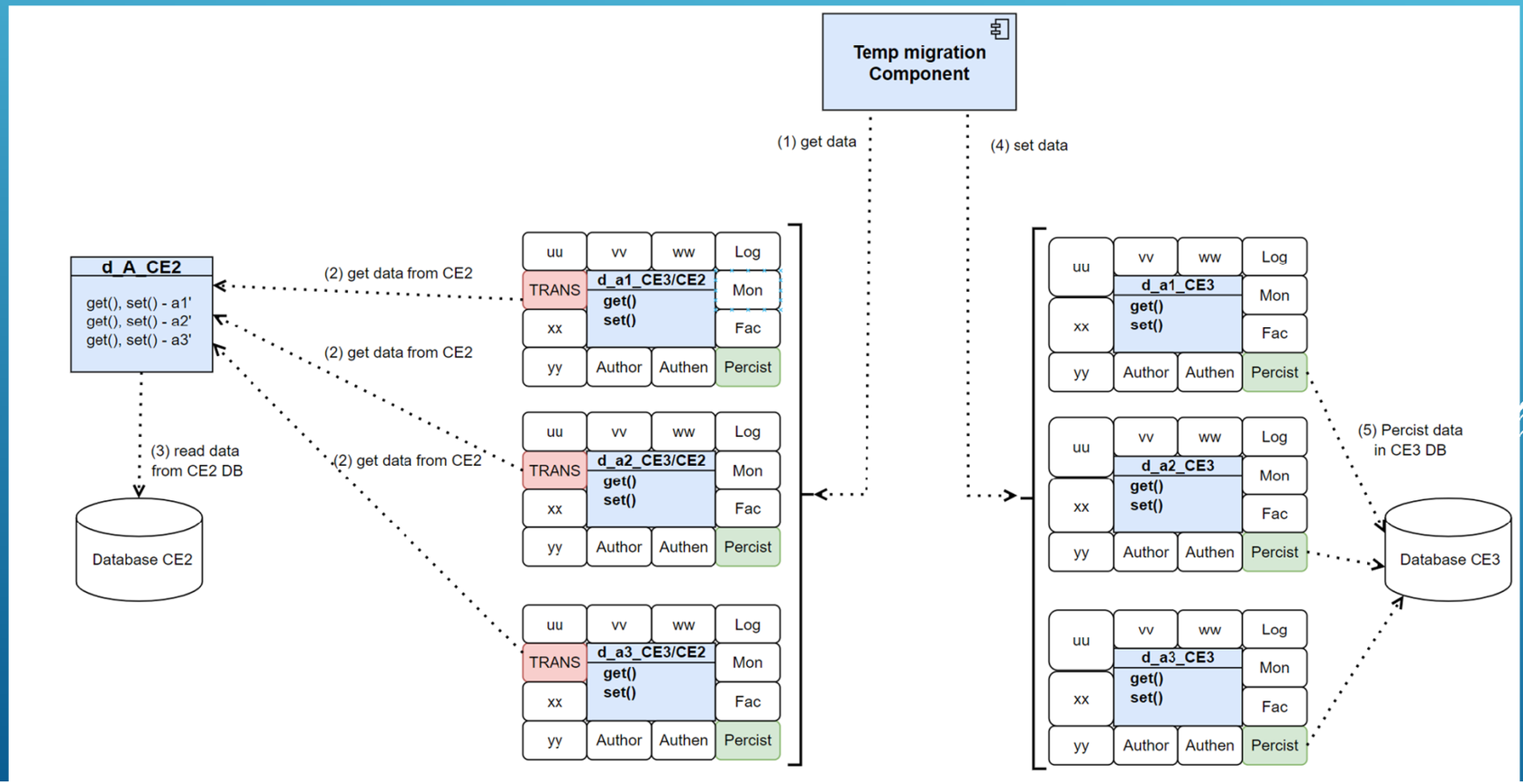
Use Case: Connecting Expertise – migration approach

Phased approach in combination with Chicken Little Migration Methodology
Improve anthropomorphism
Apply NS principles
Integration Migration in the Expansion
Remove the Integration via Expansion



Use Case: Connecting Expertise – migration approach

Integration Migration in the Expansion
Remove the Integration via Expansion



Use Case: Connecting Expertise – Value of NS

New version of CE (CE3), has a state of the art, NS compliant architecture.
CE will become, gradually, a truly agile system.

Anthropomorphic approach to naming of objects increases code readability and decreases complexity

The major drawback of the phased approach (Chicken Little) – gateway complexity – is reduced due to the integration of the gateways in the expansion templates (transformers).

Gateways can be easily removed via re-expansion, remove all traces from legacy.