University of Antwerp

Systematic Rejuvenation of a Budgeting Application over 10 years: A Case Study

Chetak Kandaswamy, Jan Verelst

Prof. Dr. Jan Verelst

Received his PhD 1999 from the University of Antwerp

University of Antwerp (UA)

- Chairman Dept. Of Management Information Systems
- Program Director Digital Business Engineering

Antwerp Management School (AMS)

- Post-experience education in the Master Enterprise IT Architecture

Technical University Prague (CTU)

- Pre-experience education on Normalized Systems

Research on Normalized Systems at UA

- Co-founded research group on NS at UA in 2006
- Co-founded NSX bv in 2011, spin-off of the University of Antwerp

On Normalized Systems

About Normalized Systems Theory (NST)

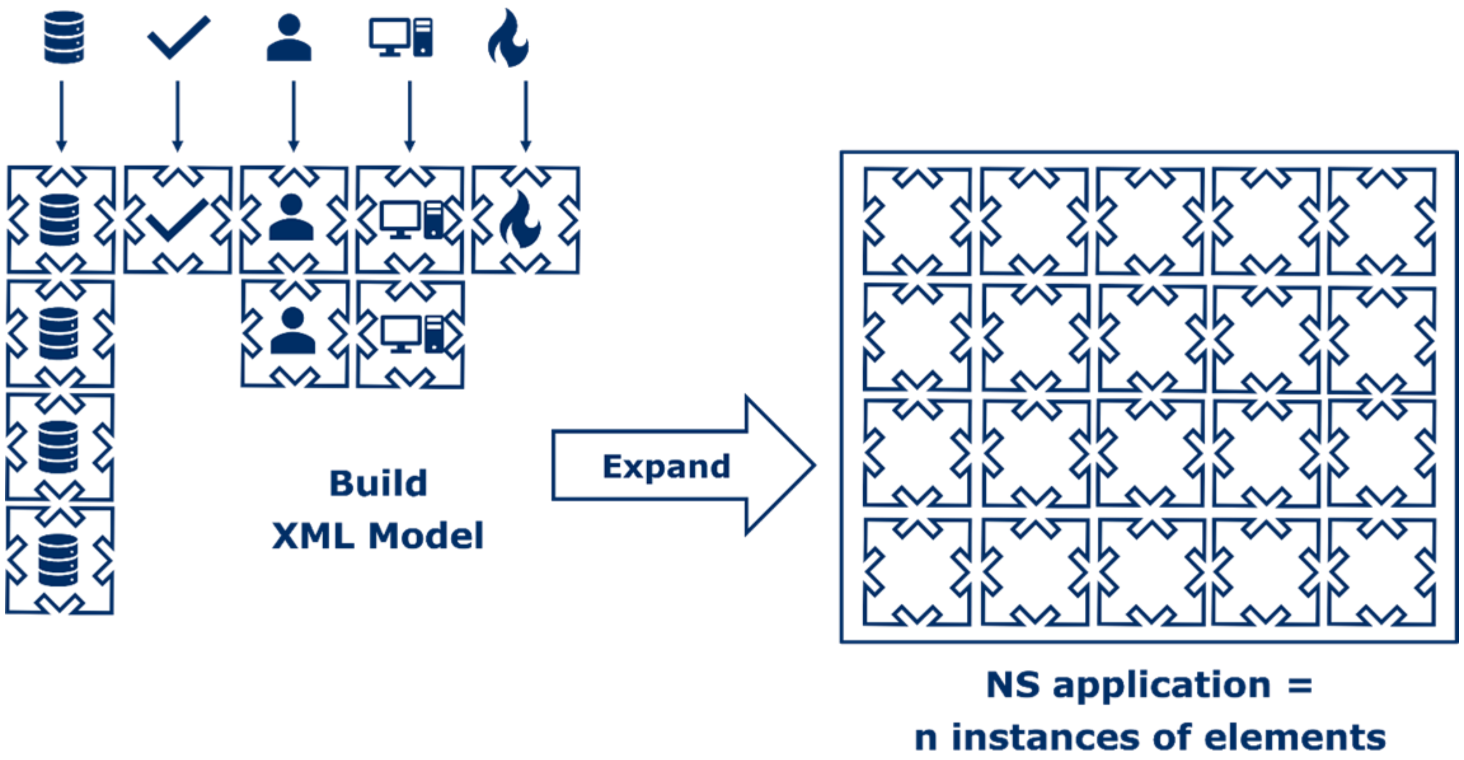
- **A theoretical framework investigating Modular Structures under Change**
  - Based on concepts from Systems Theory and Thermodynamics
    - Completely independent of any framework, programming language, package, …
  - Initial scope: Modular Structures in Software Architectures for Information Systems
  - Publications: book, >50 papers & conference proceedings, (invited) lectures at different universities…
  - Education: undergraduate, postgraduate
- **Industrial practice: substantial installed base**

NS Theorems

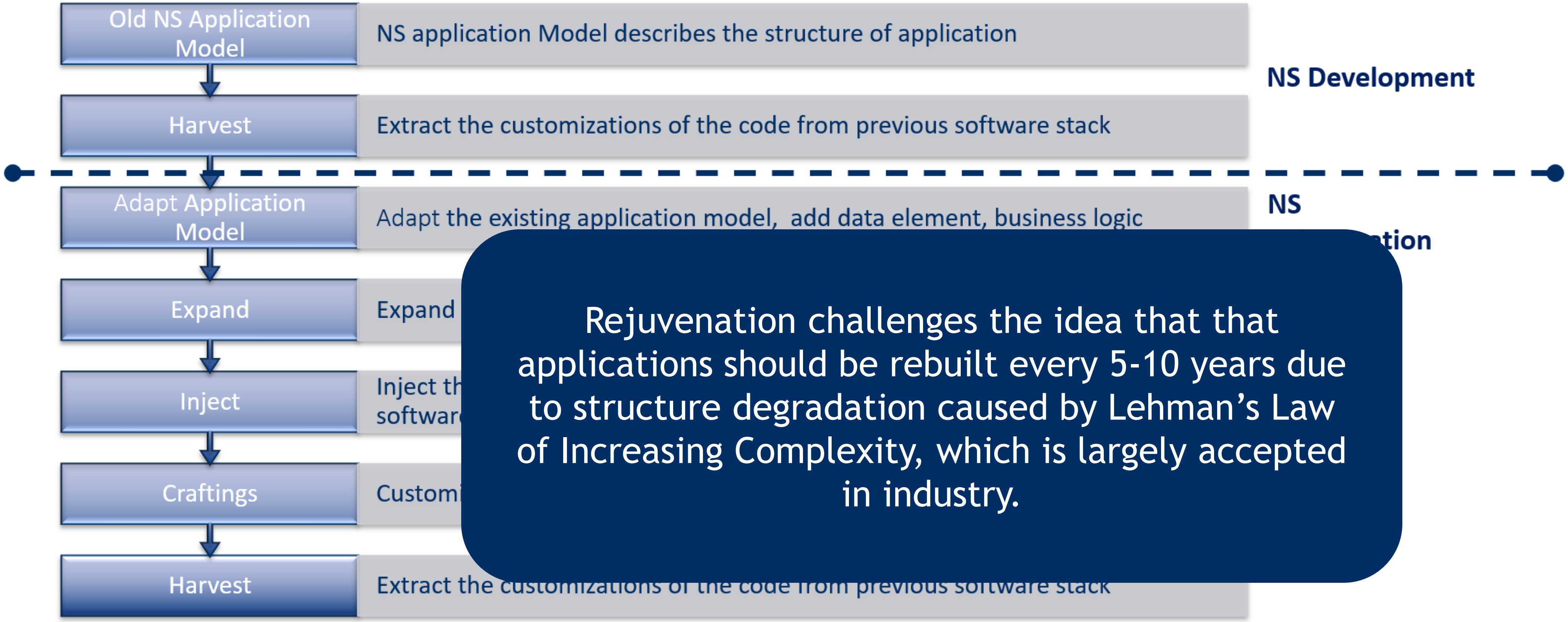
NST defines evolvable modularity as absence of so-called Combinatorial Effects (CE), which occur with every violation during development or maintenance of the following design principles:

- Separation of Concerns
- Data Version Transparency
- Action Version Transparency
- Separation of State

Expansion of NS Elements



Rejuvenation of NS Applications



Old NS Application Model — NS application Model describes the structure of application

Harvest — Extract the customizations of the code from previous software stack

NS Development

Adapt Application Model — Adapt the existing application model, add data element, business logic

Expand — Expand

Inject — Inject th softwar

Craftings — Customi

Harvest — Extract the customizations of the code from previous software stack

NS

tion

Rejuvenation challenges the idea that that applications should be rebuilt every 5-10 years due to structure degradation caused by Lehman’s Law of Increasing Complexity, which is largely accepted in industry.

Case Study:

Province of Antwerp Budgeting Application

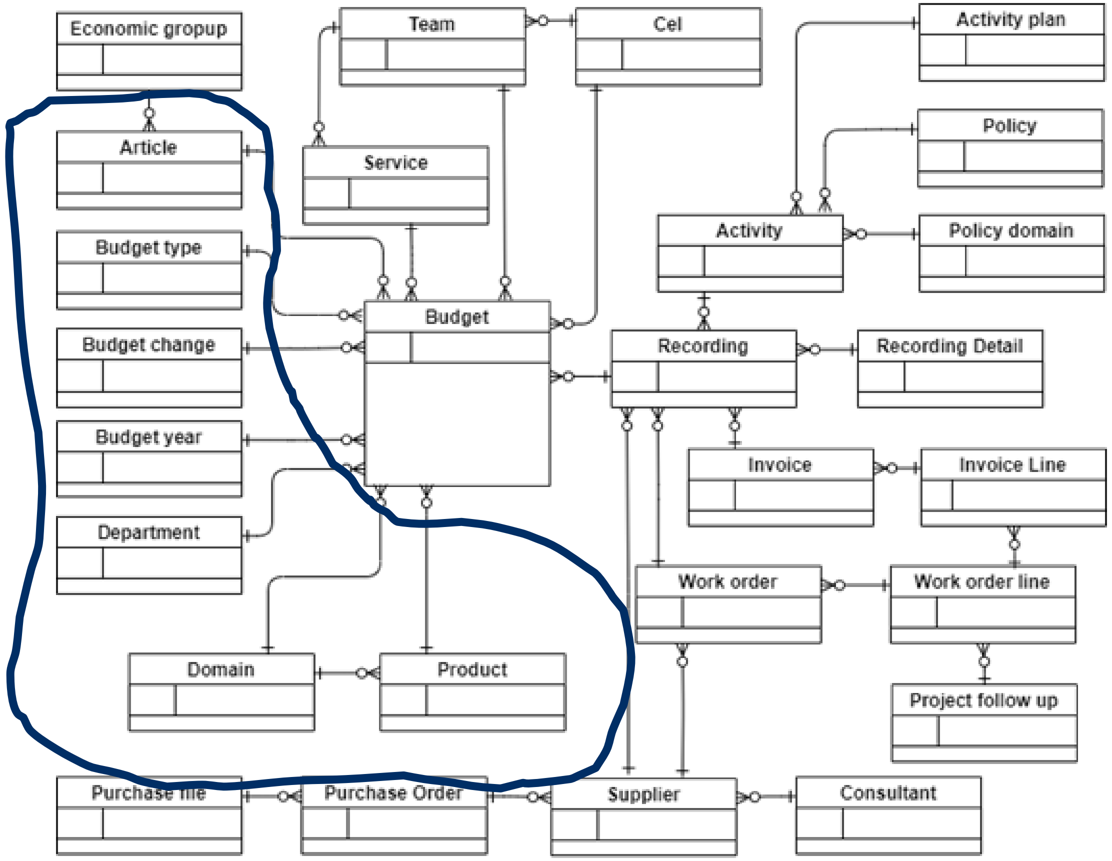
About the Case Study

- **Origin**
  - in 2012, existing ERP package was difficult to adapt to the specifics of Belgian government regulations

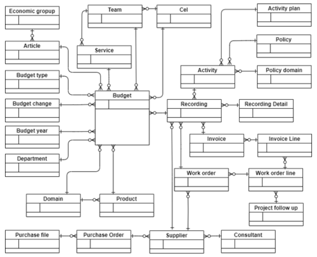
- **Goal**
  - monitor and manage its financial resources effectively,
  - ensure compliance with fiscal policies,
  - make data-driven decisions to allocate resources efficiently
  - integrate with existing financial systems, such as accounting software and ERP systems

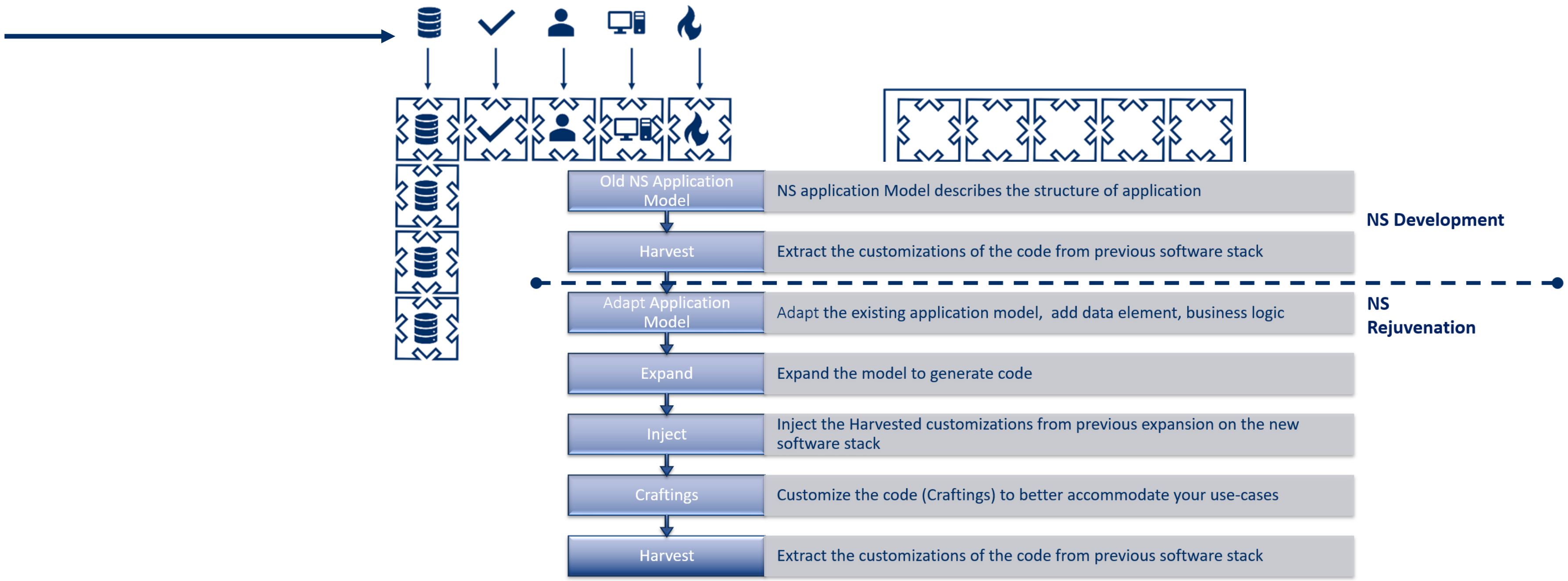
- **Functional requirements**
  - budget creation and management,
  - expense tracking and control,
  - managing different revenue streams,
  - forecasting and planning,
  - reporting and analysis,
  - compliance and audit trail,
  - integration with financial systems,
  - data security,
  - and privacy.

Data Model


Economic gropup
Team
Cel
Activity plan
Article
Service
Policy
Budget type
Activity
Policy domain
Budget
Budget change
Recording
Recording Detail
Budget year
Invoice
Invoice Line
Department
Work order
Work order line
Domain
Product
Project follow up
Purchase file
Purchase Order
Supplier
Consultant

Data Model as Input to the NS Expanders





Old NS Application Model — NS application Model describes the structure of application

Harvest — Extract the customizations of the code from previous software stack

NS Development

NS Rejuvenation

Adapt Application Model — Adapt the existing application model, add data element, business logic

Expand — Expand the model to generate code

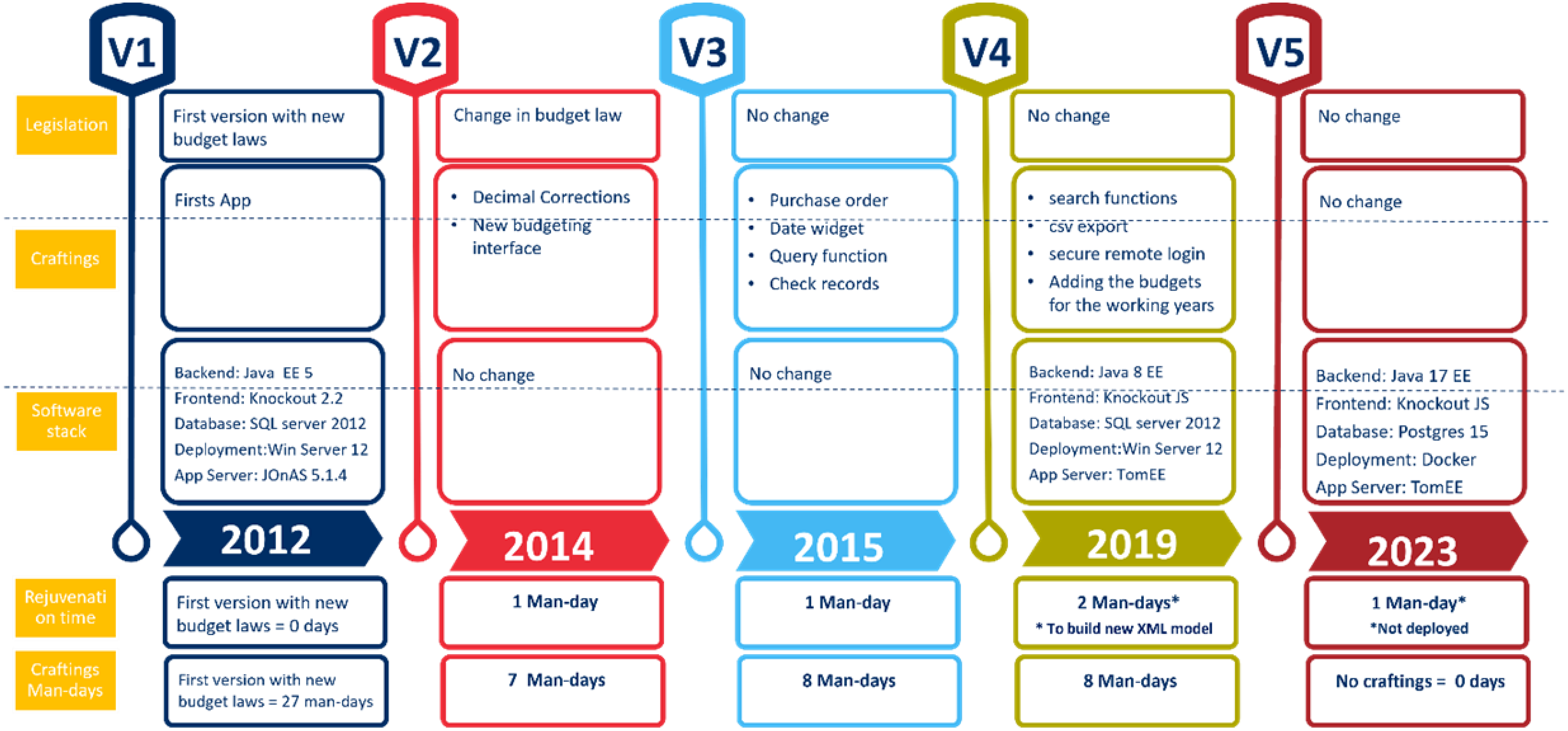
Inject — Inject the Harvested customizations from previous expansion on the new software stack

Craftings — Customize the code (Craftings) to better accommodate your use-cases

Harvest — Extract the customizations of the code from previous software stack

University of Antwerp

Rejuvenation 2012-2023



	V1	V2	V3	V4	V5
Legislation	First version with new budget laws	Change in budget law	No change	No change	No change
Craftings	Firsts App	• Decimal Corrections • New budgeting interface	• Purchase order • Date widget • Query function • Check records	• search functions • csv export • secure remote login • Adding the budgets for the working years	No change
Software stack	Backend: Java EE 5 Frontend: Knockout 2.2 Database: SQL server 2012 Deployment:Win Server 12 App Server: JOnAS 5.1.4	No change	No change	Backend: Java 8 EE Frontend: Knockout JS Database: SQL server 2012 Deployment:Win Server 12 App Server: TomEE	Backend: Java 17 EE Frontend: Knockout JS Database: Postgres 15 Deployment: Docker App Server: TomEE
	2012	2014	2015	2019	2023
Rejuvenation time	First version with new budget laws = 0 days	1 Man-day	1 Man-day	2 Man-days* * To build new XML model	1 Man-day* *Not deployed
Craftings Man-days	First version with new budget laws = 27 man-days	7 Man-days	8 Man-days	8 Man-days	No craftings = 0 days

Limitations

- Only a small CRUDS application

- The technical changes were limited in the sense that updates from technologies were required, but no major shifts to other technologies.

Voice of the Customer

- **On the advantages of Rejuvenation using NS framework**
  - "The main advantage for us was the **speed** that can be gained with the **rejuvenation** of the application. Because the process of expansion and re-injection is fully automated and fast, a new version can be put in place and the actual functionality can be tested instead of also having to validate and test the boiler-plate code."
- **On developing with or without NS**
  - "We have no real data concerning the effective difference between development with or without NS. **In my opinion**, if we did not use NS, the first change of the application in 2014 (new budgeting legislation), would have resulted in **building a new application**, instead of just rejuvenating the existing one. Such a rebuild would have probably taken 50 man-days. While with rejuvenation, we only had a few days of functional testing to do."
- **On Maintenance Cost**
  - The maintenance of 6 different applications at the Province of Antwerp built using the NS methods (including the Budgeting application) required **only 4 man-days of maintenance operations both in 2021 and in 2022 (across all 6 applications)**.

Conclusion

- **NST was published extensively in previous papers**

- **This paper documents one of the essential characteristics in NST, i.e.,**
  - a **rejuvenation** over a **period of 10 years**, with the skeleton of the application being updated to the most recent version of the underlying technologies,
  - in response to m**ultiple functional and non-functional, technical changes**
  - whereas in industry, it is largely accepted that applications should be rebuilt every 5-10 years

- **This suggests that the increased use of code generators holds significant promise for the future.**

Bibliography

- H. Mannaert, J. Verelst, and P. De Bruyn, “Normalized Systems Theory: From Foundations for Evolvable Software Toward a General Theory for Evolvable Design”, Koppa Publishing, ISBN 978-90-77160-09-1, 2016.

- [2] H. Mannaert, J. Verelst, and K. Ven, “The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability”, Science of Computer Programming, Volume 76, Issue 12, pp. 1210-1222, 2011.

- [3] P. Huysmans, G. Oorts, P. De Bruyn, H. Mannaert, and J. Verelst, “Positioning
- the normalized systems theory in a design theory framework”, Lecture notes in business information processing, Springer, ISSN 1865-1348-142, pp. 43-63, 2013.

- [4] G. Oorts, et al., “Building Evolvable Software Using Normalized Systems Theory: A Case Study”, Proceedings of the annual Hawaii international conference on system sciences, ISBN 978-1-4799-2504-9, pp. 4760-4769, 2014.