Managing structural and behavioral evolution in relational database:

Application of Software Engineering techniques

Ph.D defense

Julien Delplanque

julien.delplanque@inria.fr

Université de Lille

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Inria
inventeurs du monde numérique

RMod

Roadmap

- Context & State of the Art
- Motivation
- A Behavior-Aware Meta-Model for Relational Databases
- Identifying Quality Issues in Relational Databases
- Recommendations for Evolving Relational Databases
- Conclusion

Roadmap

- **Context & State of the Art**

- Motivation
- A Behavior-Aware Meta-Model for Relational Databases
- Identifying Quality Issues in Relational Databases
- Recommendations for Evolving Relational Databases
- Conclusion

Structure and behavior in relational databases

Structural entities describe how data are structured

- Columns
- Tables
- Primary Keys
- Foreign Keys
- ...

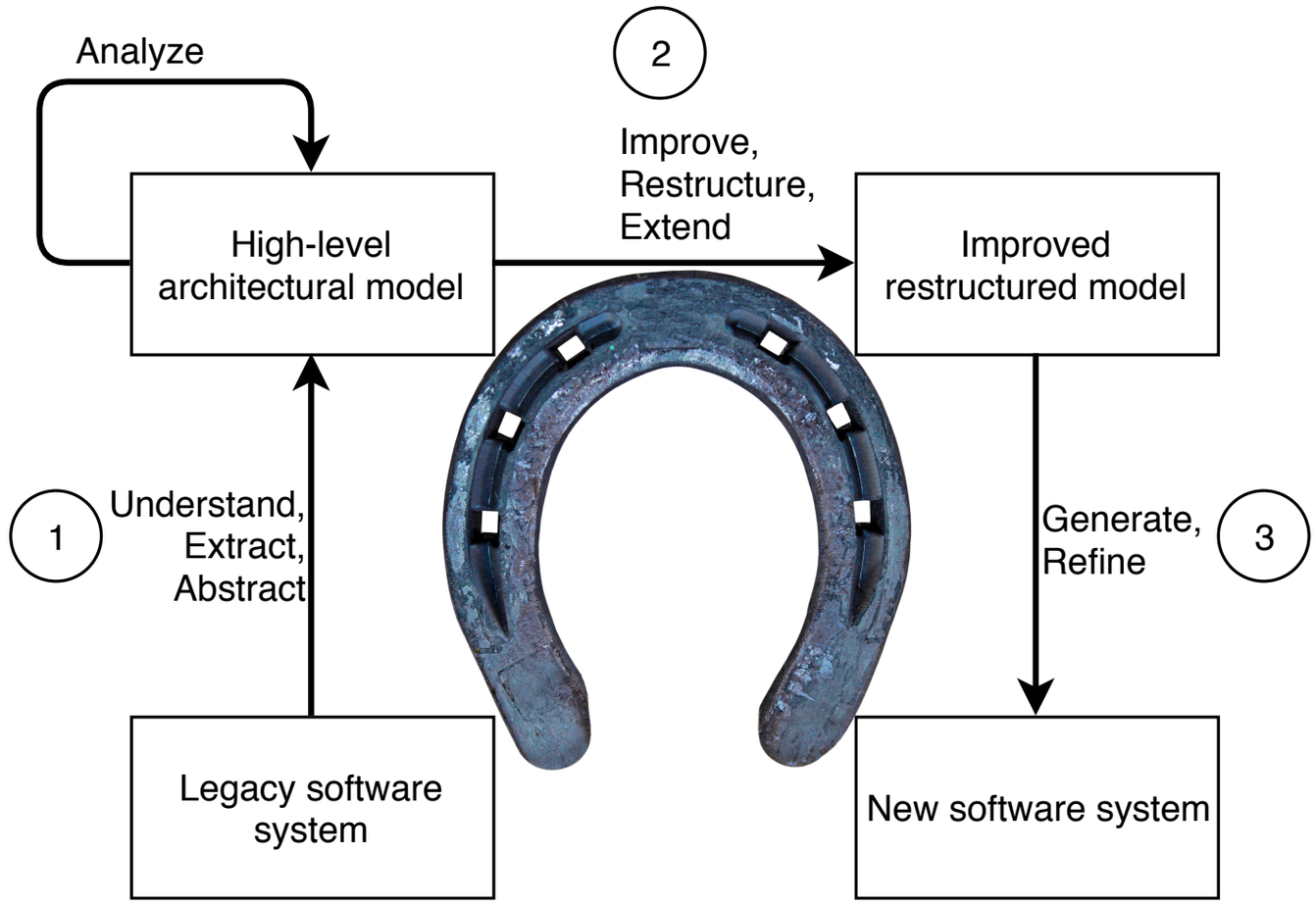
Behavioral entities describe computation

- Views
- Stored procedures
- Triggers
- ...

Software Evolution

E-type software: *“Programs that mechanize a human or societal activity.”* [Lehman 1980]

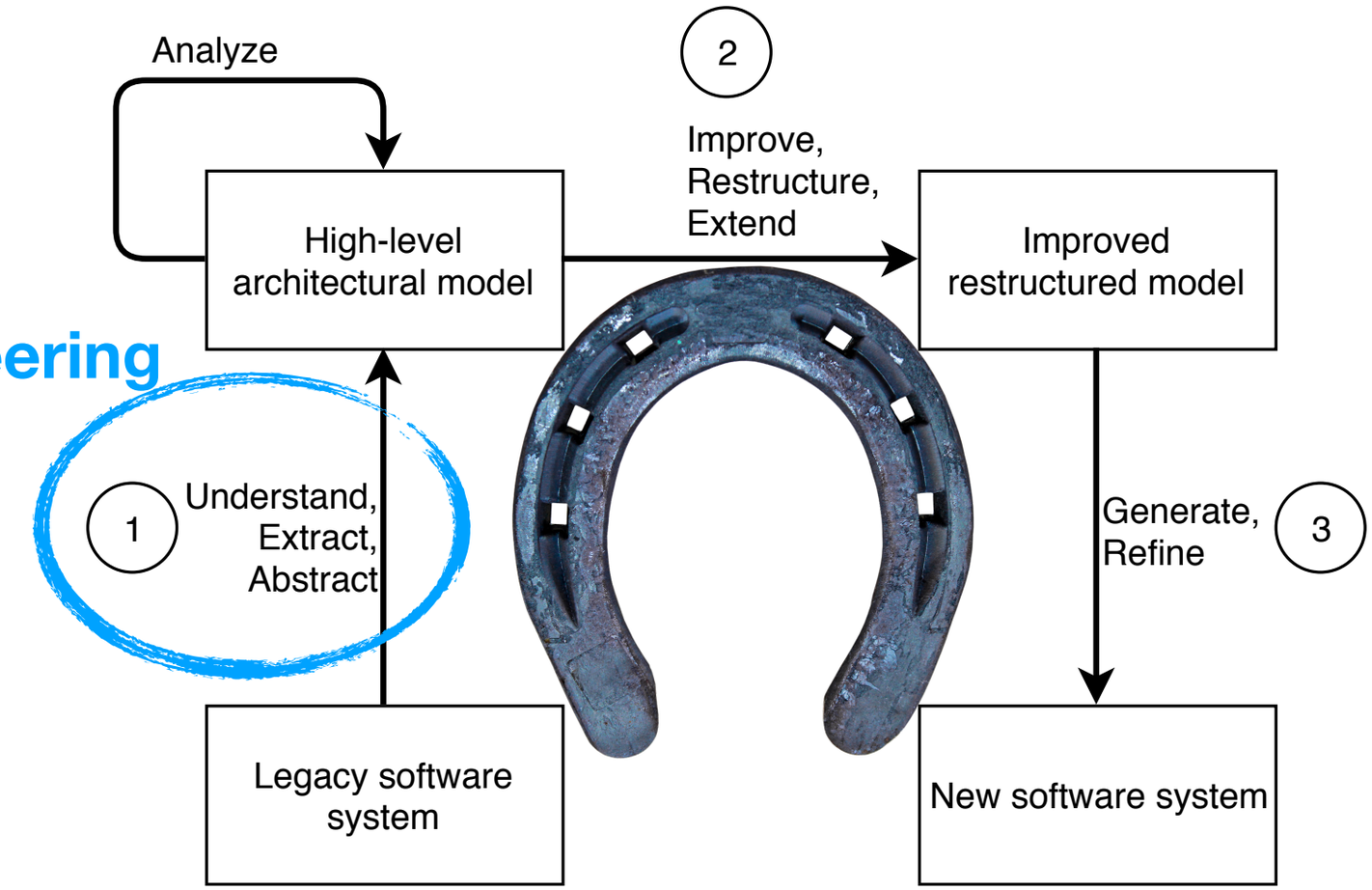
Software Re-engineering



The horseshoe process [Kazman 1998]

Software Re-engineering

Reverse Engineering



The horseshoe process [Kazman 1998]

Approaches to Reverse-Engineer Relational Databases

Review of the literature based on 3 criteria

Approach	Crit. 1.	Crit. 2.	Crit. 3.
[Markowitz 1990]	Struct.	EER	Auto.
[Castellanos 1993]	Struct.	BLOOM	Human
[Shoval 1993]	Struct.	NBR	Human
[Premerlani 1993]	Struct.	OMT	Human
[Chiang 1994]	Struct.	EER	Auto.
[Petit 1994]	Struct.	EER	Auto.
[Henrard 1998]	Struct.	EER	Human
[Polo 2002]	Struct./Beha.	Custom meta-model	Auto.
[Cleve 2006]	Struct./Ext.	SDG	Auto.
[Cleve 2008]	Struct./Ext.	Not specified	Auto.
[Yeh 2008]	Struct.	EER	Auto.
[Wang 2009]	Struct.	Custom meta-model	Auto.
[Papastefanatos 2008]	Struct./Beha.	Directed graph	Auto.
[Alalfi 2009]	Ext.	Custom meta-model	Auto.
[Cleve 2011a]	Struct./Ext.	Not specified	Auto.
[Meurice 2016a]	Struct./Ext.	ER	Auto.

Approaches to Reverse-Engineer Relational Databases

Crit. 1. For which kind of entities does the approach gather information to build a model of the database?

Approach	Crit. 1.	Crit. 2.	Crit. 3.
[Markowitz 1990]	Struct.	EER	Auto.
[Castellanos 1993]	Struct.	BLOOM	Human
[Shoval 1993]	Struct.	NBR	Human
[Premerlani 1993]	Struct.	OMT	Human
[Chiang 1994]	Struct.	EER	Auto.
[Petit 1994]	Struct.	EER	Auto.
[Henrard 1998]	Struct.	EER	Human
[Polo 2002]	Struct./Beha.	Custom meta-model	Auto.
[Cleve 2006]	Struct./Ext.	SDG	Auto.
[Cleve 2008]	Struct./Ext.	Not specified	Auto.
[Yeh 2008]	Struct.	EER	Auto.
[Wang 2009]	Struct.	Custom meta-model	Auto.
[Papastefanatos 2008]	Struct./Beha.	Directed graph	Auto.
[Alalfi 2009]	Ext.	Custom meta-model	Auto.
[Cleve 2011a]	Struct./Ext.	Not specified	Auto.
[Meurice 2016a]	Struct./Ext.	ER	Auto.

Approaches to Reverse-Engineer Relational Databases

Crit. 1. For which kind of entities does the approach gather information to build a model of the database?

Approach	Crit. 1.	Crit. 2.	Crit. 3.
[Markowitz 1990]	Struct.	EER	Auto.
[Castellanos 1993]	Struct.	BLOOM	Human
[Shoval 1993]	Struct.	NBR	Human
[Premerlani 1993]	Struct.	OMT	Human
[Chiang 1994]	Struct.	EER	Auto.
[Petit 1994]	Struct.	EER	Auto.
[Henrard 1998]	Struct.	EER	Human
[Polo 2002]	Struct./Beha.	Custom meta-model	Auto.
[Cleve 2006]	Struct./Ext.	SDG	Auto.
[Cleve 2008]	Struct./Ext.	Not specified	Auto.
[Yeh 2008]	Struct.	EER	Auto.
[Wang 2009]	Struct.	Custom meta-model	Auto.
[Papastefanatos 200	Struct./Beha.	Directed graph	Auto.
[Alalfi 2009]	Ext.	Custom meta-model	Auto.
[Cleve 2011a]	Struct./Ext.	Not specified	Auto.
[Meurice 2016a]	Struct./Ext.	ER	Auto.

Approaches to Reverse-Engineer Relational Databases

Crit. 1. For which kind of entities does the approach gather information to build a model of the database?

Approach	Crit. 1.	Crit. 2.	Crit. 3.
[Markowitz 1990]	Struct.	EER	Auto.
[Castellanos 1993]	Struct.	BLOOM	Human
[Shoval 1993]	Struct.	NBR	Human
[Premerlani 1993]	Struct.	OMT	Human
[Chiang 1994]	Struct.	EER	Auto.
[Petit 1994]	Struct.	EER	Auto.
[Henrard 1998]	Struct.	EER	Human
[Polo 2002]	Struct./Beha.	Custom meta-model	Auto.
[Cleve 2006]	Struct./Ext.	SDG	Auto.
[Cleve 2008]	Struct./Ext.	Not specified	Auto.
[Yeh 2008]	Struct.	EER	Auto.
[Wang 2009]	Struct.	Custom meta-model	Auto.
[Papastefanatos 2008]	Struct./Beha.	Directed graph	Auto.
[Alalfi 2009]	Ext.	Custom meta-model	Auto.
[Cleve 2011a]	Struct./Ext.	Not specified	Auto.
[Meurice 2016a]	Struct./Ext.	ER	Auto.

Approaches to Reverse-Engineer Relational Databases

Crit. 2. What formalism is used to model the database and programs using it?

Approach	Crit. 1.	Crit. 2.	Crit. 3.
[Markowitz 1990]	Struct.	EER	Auto.
[Castellanos 1993]	Struct.	BLOOM	Human
[Shoval 1993]	Struct.	NBR	Human
[Premerlani 1993]	Struct.	OMT	Human
[Chiang 1994]	Struct.	EER	Auto.
[Petit 1994]	Struct.	EER	Auto.
[Henrard 1998]	Struct.	EER	Human
[Polo 2002]	Struct./Beha.	Custom meta-model	Auto.
[Cleve 2006]	Struct./Ext.	SDG	Auto.
[Cleve 2008]	Struct./Ext.	Not specified	Auto.
[Yeh 2008]	Struct.	EER	Auto.
[Wang 2009]	Struct.	Custom meta-model	Auto.
[Papastefanatos 2008]	Struct./Beha.	Directed graph	Auto.
[Alalfi 2009]	Ext.	Custom meta-model	Auto.
[Cleve 2011a]	Struct./Ext.	Not specified	Auto.
[Meurice 2016a]	Struct./Ext.	ER	Auto.

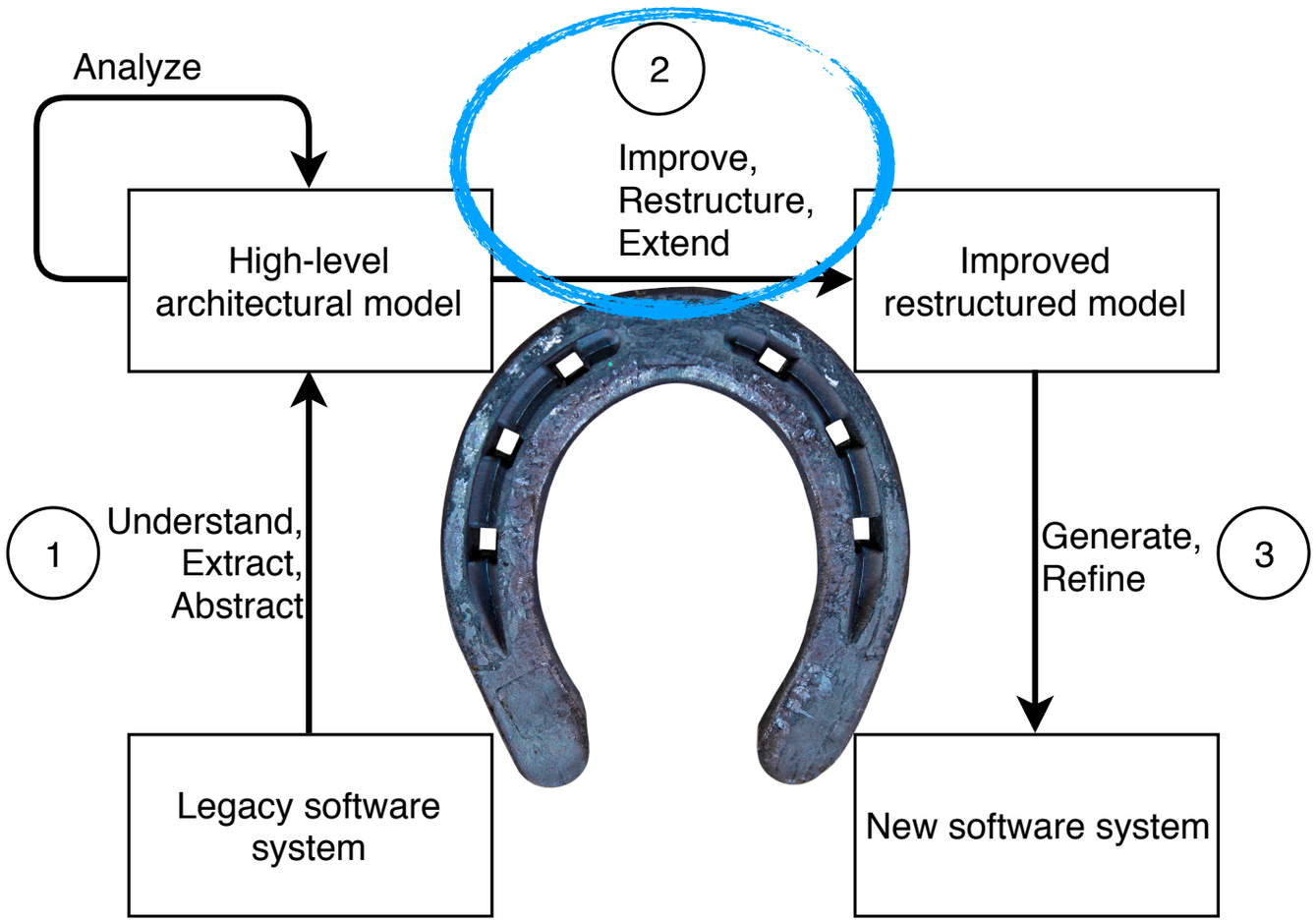
Approaches to Reverse-Engineer Relational Databases

Crit. 3. Is there a human intervention during the model importation or is the approach automatic?

Approach	Crit. 1.	Crit. 2.	Crit. 3.
[Markowitz 1990]	Struct.	EER	Auto.
[Castellanos 1993]	Struct.	BLOOM	Human
[Shoval 1993]	Struct.	NBR	Human
[Premerlani 1993]	Struct.	OMT	Human
[Chiang 1994]	Struct.	EER	Auto.
[Petit 1994]	Struct.	EER	Auto.
[Henrard 1998]	Struct.	EER	Human
[Polo 2002]	Struct./Beha.	Custom meta-model	Auto.
[Cleve 2006]	Struct./Ext.	SDG	Auto.
[Cleve 2008]	Struct./Ext.	Not specified	Auto.
[Yeh 2008]	Struct.	EER	Auto.
[Wang 2009]	Struct.	Custom meta-model	Auto.
[Papastefanatos 2008]	Struct./Beha.	Directed graph	Auto.
[Alalfi 2009]	Ext.	Custom meta-model	Auto.
[Cleve 2011a]	Struct./Ext.	Not specified	Auto.
[Meurice 2016a]	Struct./Ext.	ER	Auto.

Software Re-engineering

Change Impact Analysis



The horseshoe process [Kazman 1998]

Software Change Impact Analysis

“[...] the activity of identifying what to modify to accomplish a change, or of identifying the potential consequences of a change” [Arnold 1996]

Approaches for Relational Database Impact Analysis Approaches

Review of the literature based on 5 criteria

Approach	Crit. 1. Initial change	Crit. 2. Impacted entities	Crit. 3. Change proposition	Crit. 4. Script generation	Crit. 5. Schema consistency
[Sjøberg 1993]	Struc.	Ext.	✗	✗	✗
[Haraty 2002]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2006]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2009]	Struc./Beha.	Ext.	✗	✗	✗
[Maule 2008]	Struc./Beha.	Ext.	✗	✗	✗
[Curino 2008]	Struc.	Ext.	✗	✓	✗
[Papastefanatos 2008]	Struc./Beha.	Struc./Beha.	✓	✗	✗
[Liu 2011]	Struc.	Ext.	✗	✗	✗
[Meurice 2016a]	Struc.	Ext.	✓	✗	✗

Approaches for Relational Database Impact Analysis Approaches

Crit. 1. On which kind of entities can the approach specify an initial change?

Approach	Crit. 1. Initial change	Crit. 2. Impacted entities	Crit. 3. Change proposition	Crit. 4. Script generation	Crit. 5. Schema consistency
[Sjøberg 1993]	Struc.	Ext.	✗	✗	✗
[Haraty 2002]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2006]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2009]	Struc./Beha.	Ext.	✗	✗	✗
[Maule 2008]	Struc./Beha.	Ext.	✗	✗	✗
[Curino 2008]	Struc.	Ext.	✗	✓	✗
[Papastefanatos 2008]	Struc./Beha.	Struc./Beha.	✓	✗	✗
[Liu 2011]	Struc.	Ext.	✗	✗	✗
[Meurice 2016a]	Struc.	Ext.	✓	✗	✗

Approaches for Relational Database Impact Analysis Approaches

Crit. 2. On which kind of entities can the approach compute the impact of a change?

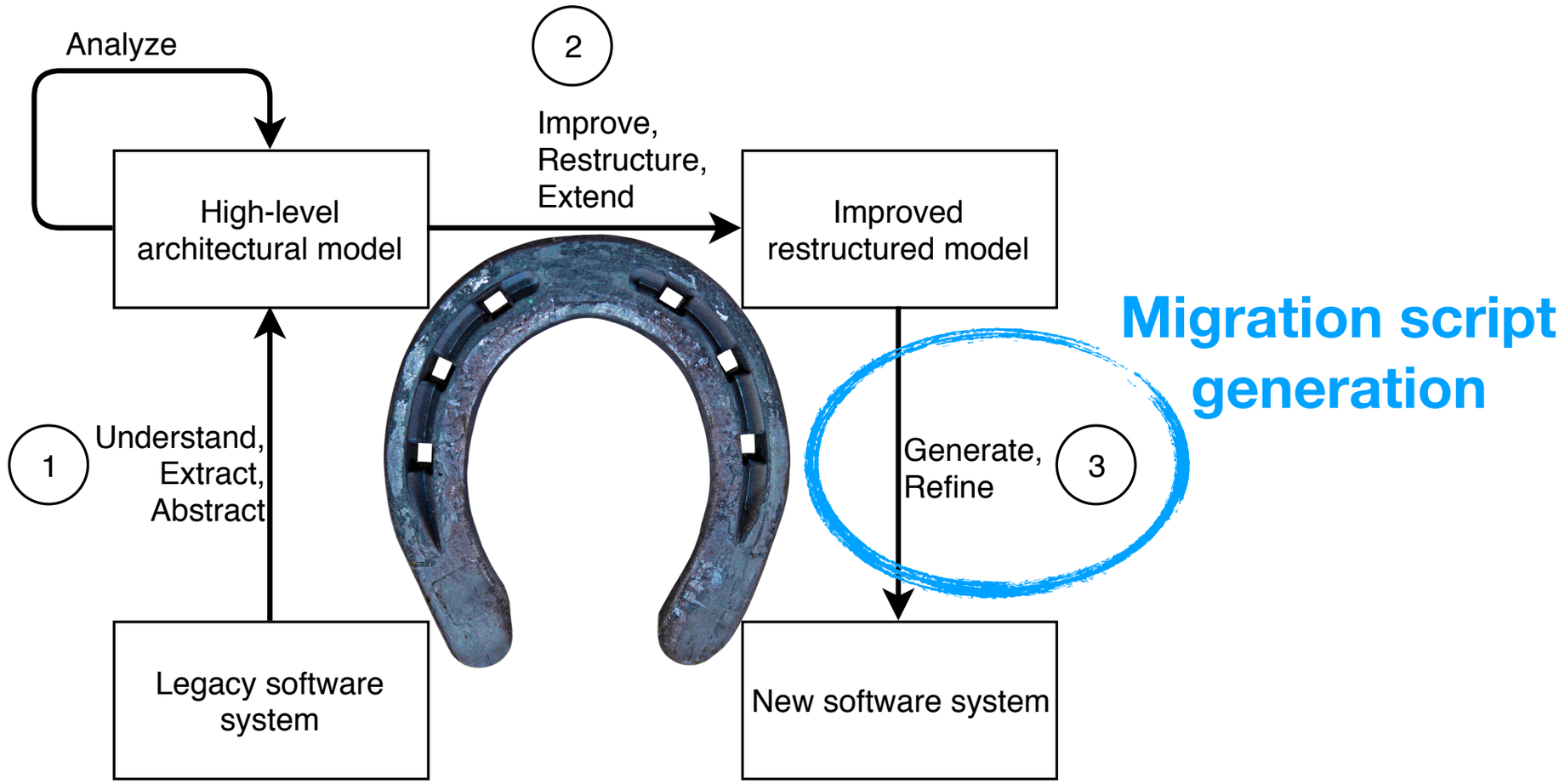
Approach	Crit. 1. Initial change	Crit. 2. Impacted entities	Crit. 3. Change proposition	Crit. 4. Script generation	Crit. 5. Schema consistency
[Sjøberg 1993]	Struc.	Ext.	✗	✗	✗
[Haraty 2002]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2006]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2009]	Struc./Beha.	Ext.	✗	✗	✗
[Maule 2008]	Struc./Beha.	Ext.	✗	✗	✗
[Curino 2008]	Struc.	Ext.	✗	✓	✗
[Papastefanatos 2008]	Struc./Beha.	Struc./Beha.	✓	✗	✗
[Liu 2011]	Struc.	Ext.	✗	✗	✗
[Meurice 2016a]	Struc.	Ext.	✓	✗	✗

Approaches for Relational Database Impact Analysis Approaches

Crit. 3. Does the approach propose changes to accommodate impacted entities with the initial change?

Approach	Crit. 1. Initial change	Crit. 2. Impacted entities	Crit. 3. Change proposition	Crit. 4. Script generation	Crit. 5. Schema consistency
[Sjøberg 1993]	Struc.	Ext.	✗	✗	✗
[Haraty 2002]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2006]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2009]	Struc./Beha.	Ext.	✗	✗	✗
[Maule 2008]	Struc./Beha.	Ext.	✗	✗	✗
[Curino 2008]	Struc.	Ext.	✗	✓	✗
[Papastefanatos 2008]	Struc./Beha.	Struc./Beha.	✓	✗	✗
[Liu 2011]	Struc.	Ext.	✗	✗	✗
[Meurice 2016a]	Struc.	Ext.	✓	✗	✗

Software Re-engineering



The horseshoe process [Kazman 1998]

Approaches for Relational Database Impact Analysis Approaches

Crit. 4. Can the approach generate a script to evolve impacted entities?

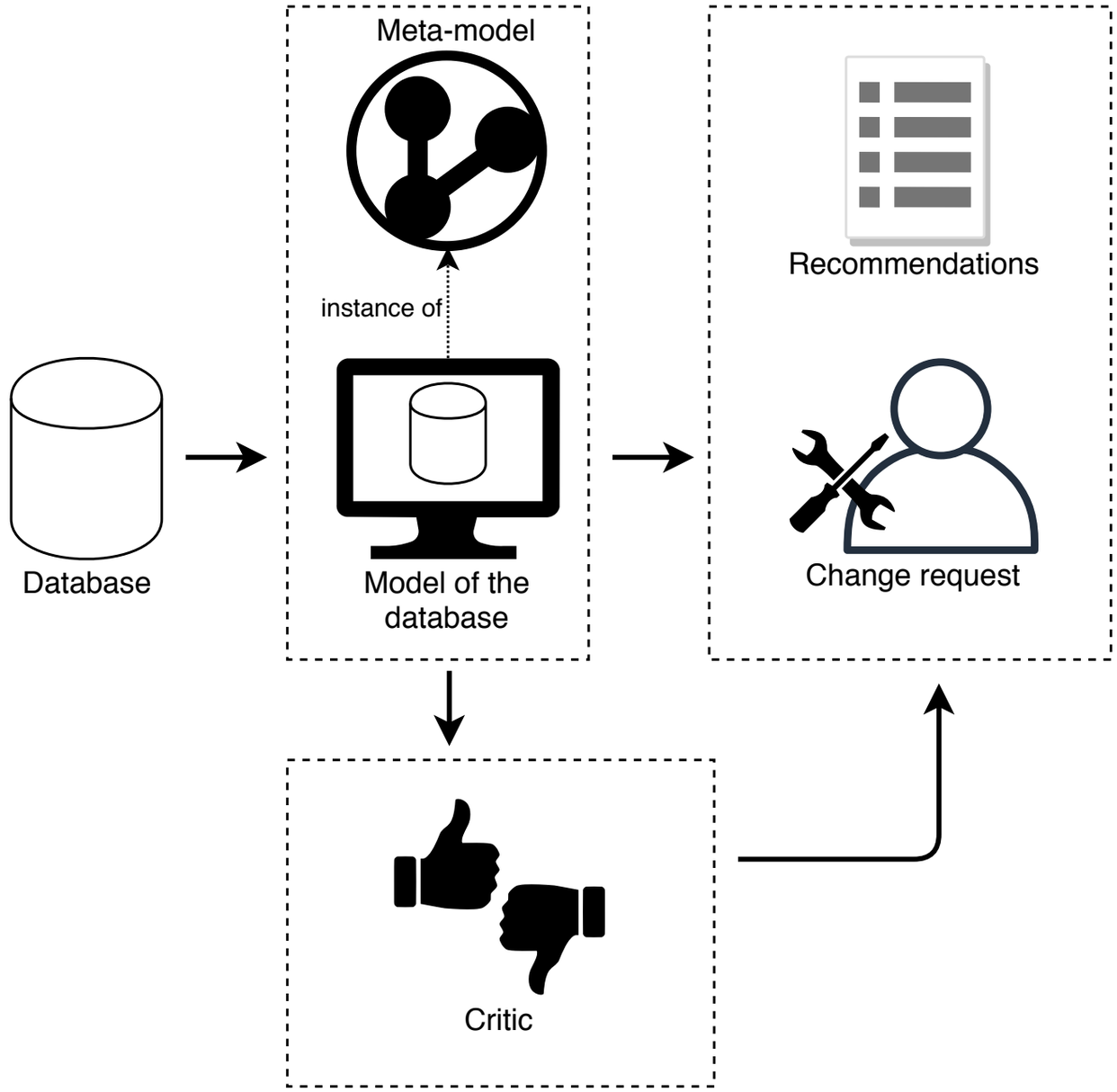
Approach	Crit. 1. Initial change	Crit. 2. Impacted entities	Crit. 3. Change proposition	Crit. 4. Script generation	Crit. 5. Schema consistency
[Sjøberg 1993]	Struc.	Ext.	✗	✗	✗
[Haraty 2002]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2006]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2009]	Struc./Beha.	Ext.	✗	✗	✗
[Maule 2008]	Struc./Beha.	Ext.	✗	✗	✗
[Curino 2008]	Struc.	Ext.	✗	✓	✗
[Papastefanatos 2008]	Struc./Beha.	Struc./Beha.	✓	✗	✗
[Liu 2011]	Struc.	Ext.	✗	✗	✗
[Meurice 2016a]	Struc.	Ext.	✓	✗	✗

Approaches for Relational Database Impact Analysis Approaches

Crit. 5. If a script is generated, does it handle database schema consistency?

Approach	Crit. 1. Initial change	Crit. 2. Impacted entities	Crit. 3. Change proposition	Crit. 4. Script generation	Crit. 5. Schema consistency
[Sjøberg 1993]	Struc.	Ext.	✗	✗	✗
[Haraty 2002]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2006]	Struc./Beha.	Ext.	✗	✗	✗
[Gardikiotis 2009]	Struc./Beha.	Ext.	✗	✗	✗
[Maule 2008]	Struc./Beha.	Ext.	✗	✗	✗
[Curino 2008]	Struc.	Ext.	✗	✓	✗
[Papastefanatos 2008]	Struc./Beha.	Struc./Beha.	✓	✗	✗
[Liu 2011]	Struc.	Ext.	✗	✗	✗
[Meurice 2016a]	Struc.	Ext.	✓	✗	✗

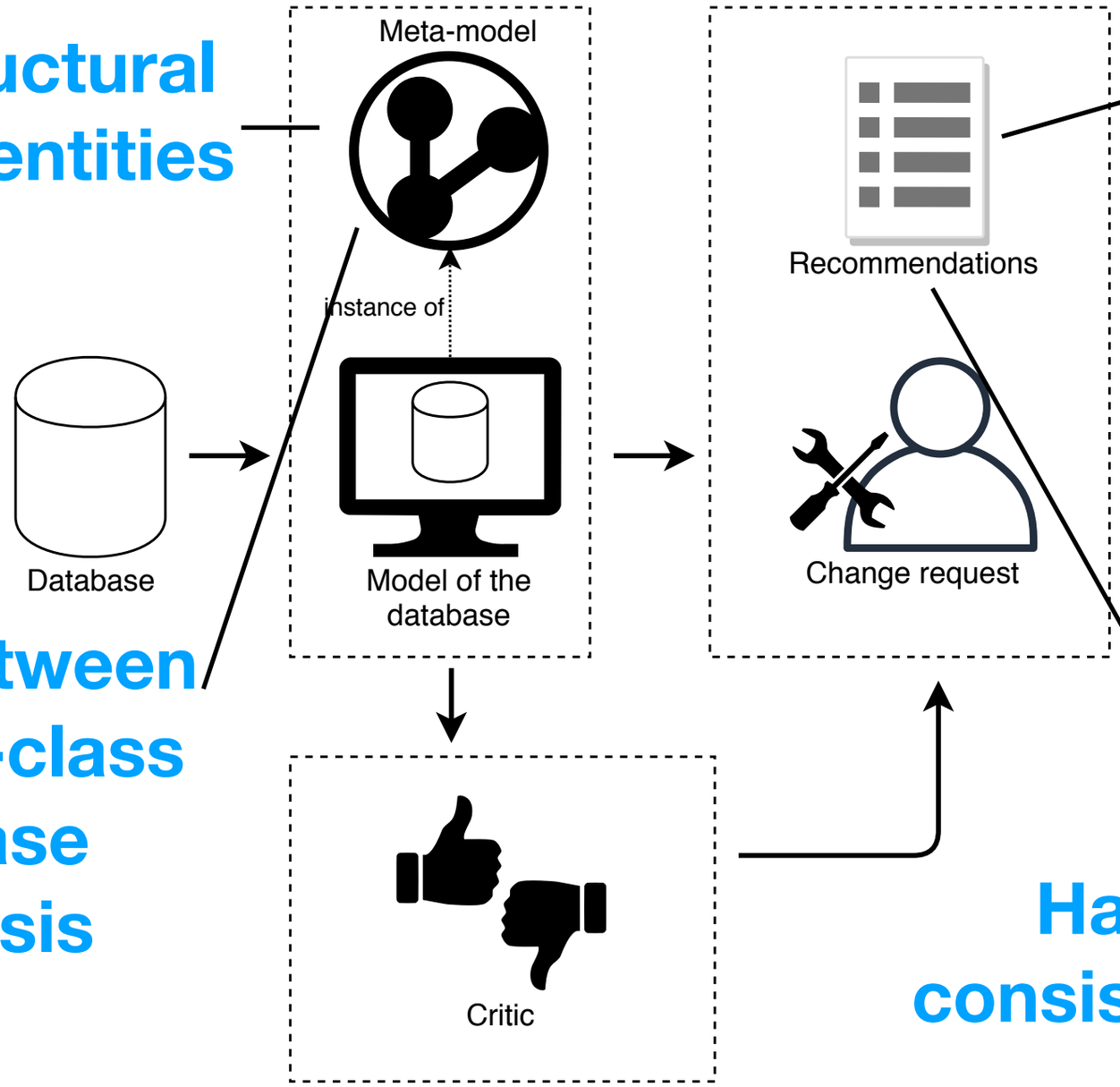
Our approach: Software Engineering for Relational Databases



Position compared to the literature

Model both structural and behavioral entities

Relationship between entities as first-class entities to ease impact analysis



Generate patch to migrate the database

Handle schema consistency constraint

Roadmap

- Context & State of the Art

- ▸ **Motivation**

- A Behavior-Aware Meta-Model for Relational Databases
- Identifying Quality Issues in Relational Databases
- Recommendations for Evolving Relational Databases
- Conclusion

Motivation

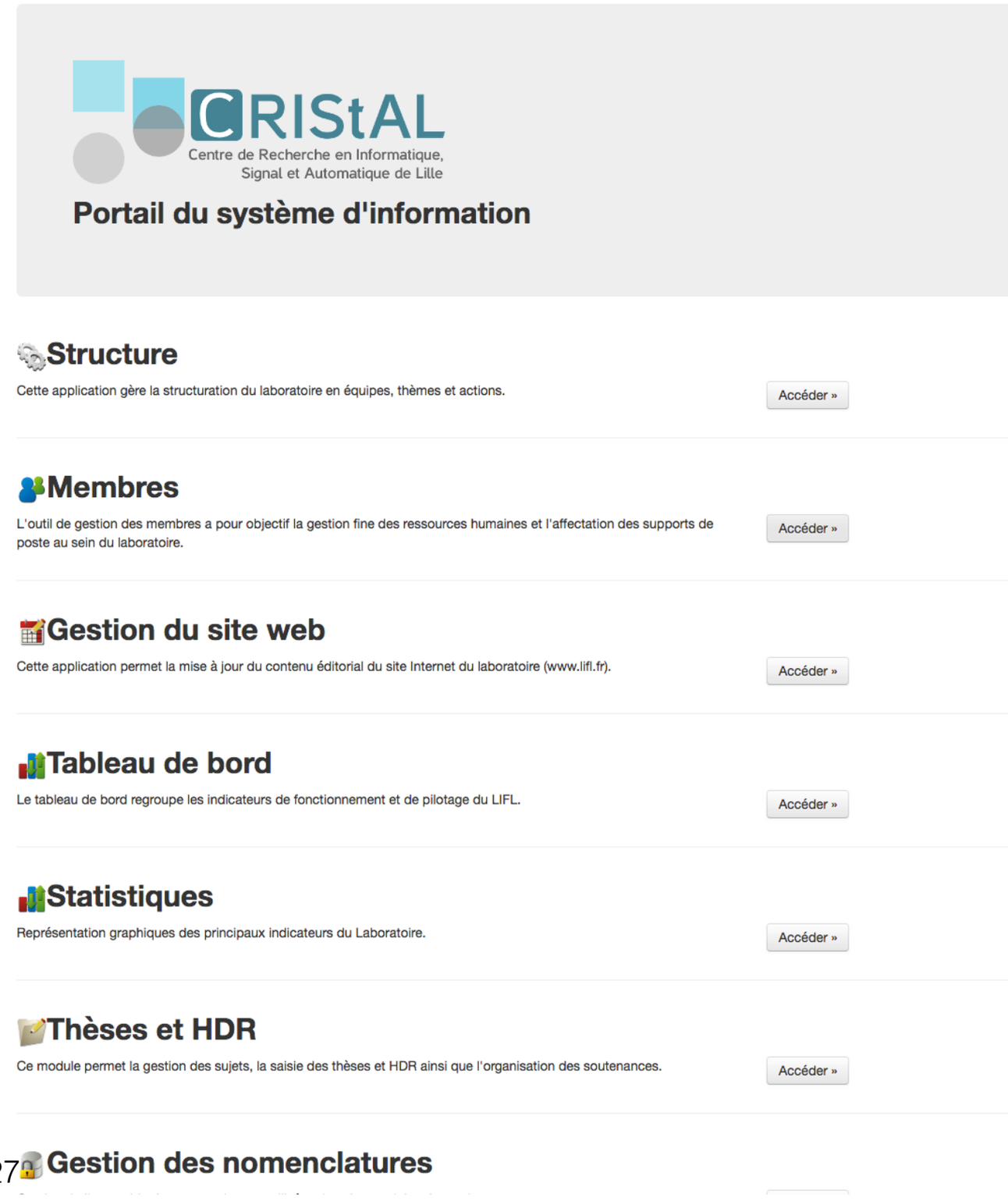
Relational Database Schema Evolution: An Industrial Case Study. ICSME’18

AppSI database overview

- Members, teams, funding support, etc. management in laboratories of our university;

- Used by software systems written in different programming languages;

- Used in multiple laboratories at the university.



An evolution of AppSI

person
id : serial (PK)
uid : varchar
email : varchar
...

Before the modification,

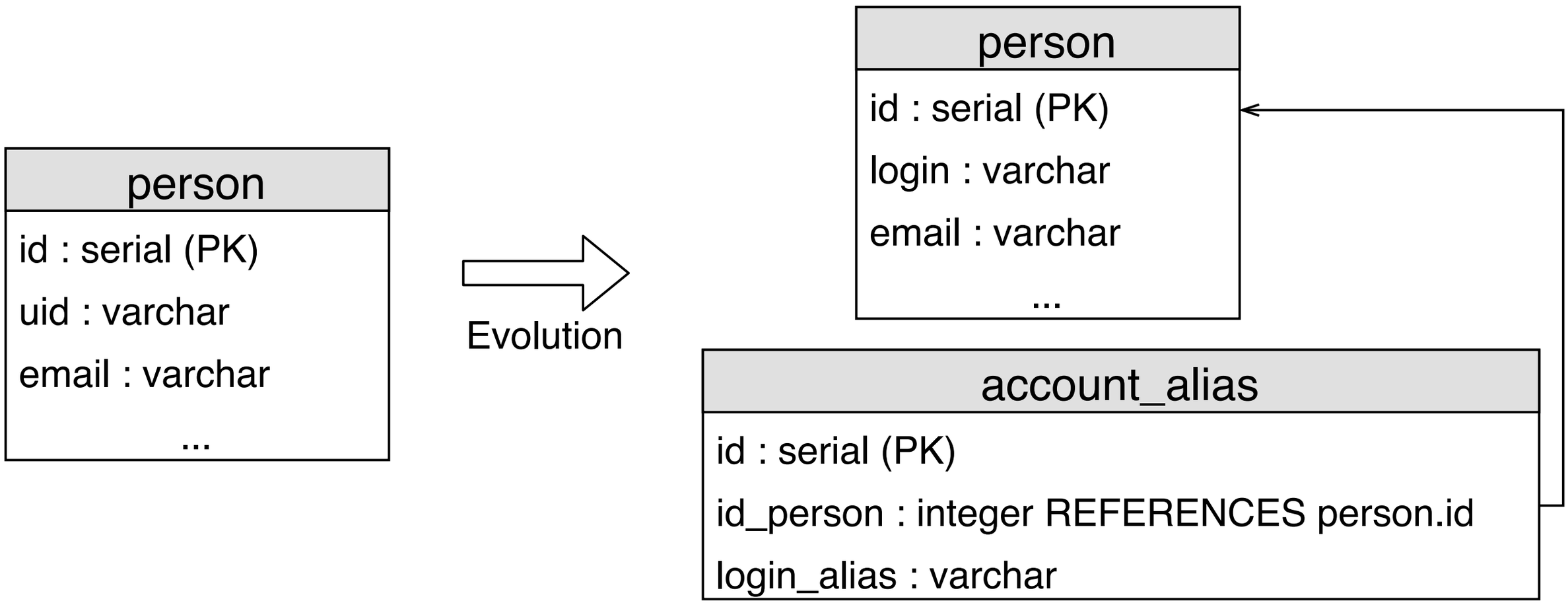
- It has a primary key (*id column*)
- It stores the LDAP identifiers in the *uid* column

New LDAP schema

- Users may have multiple identifiers
- *uid* attribute has been renamed into *login*

Consequently, *uid* column of *person* needs to be renamed into *login* **to enforce consistency** between **LDAP** and **AppSI** schemas.

An evolution of AppSI



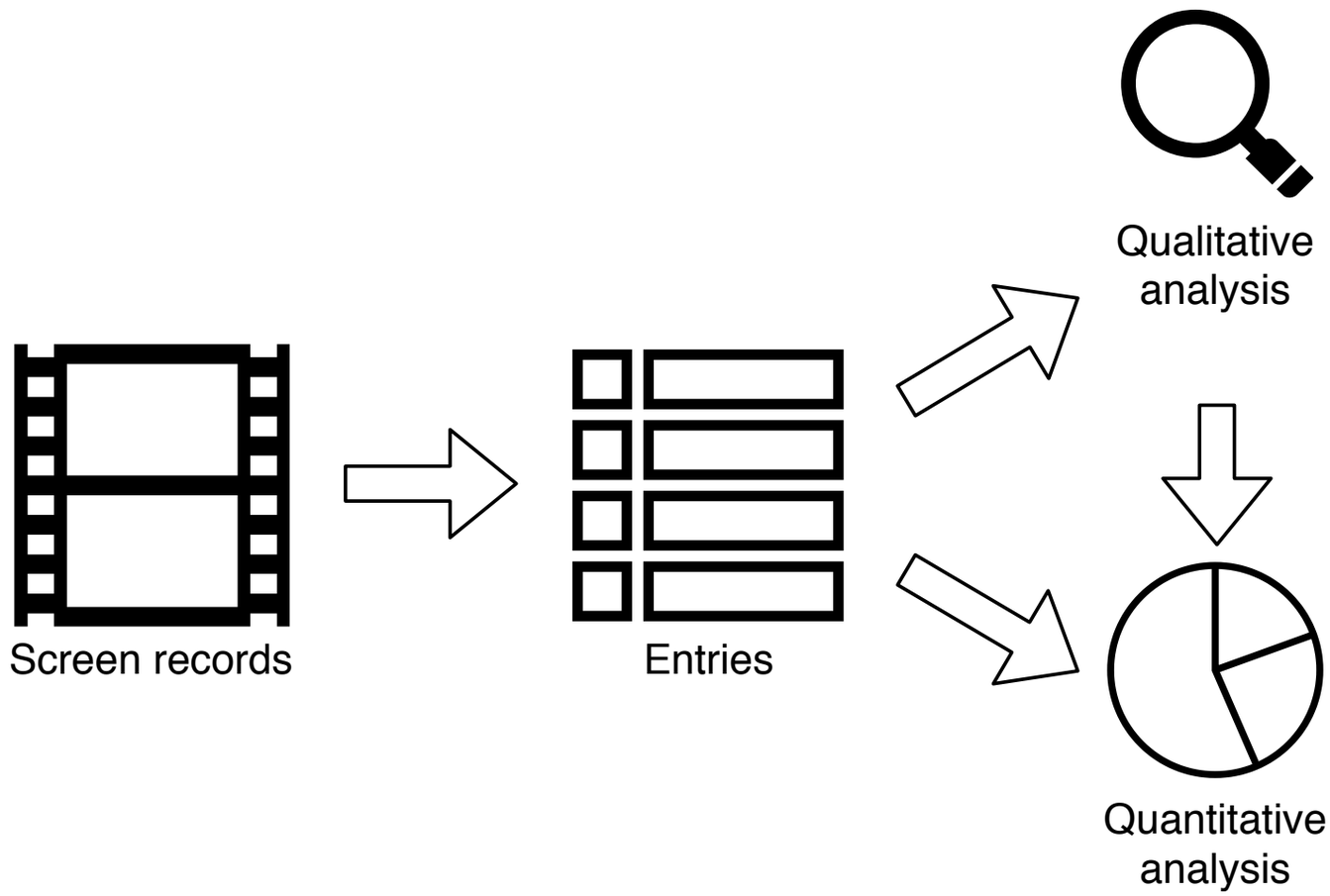
After the modification,

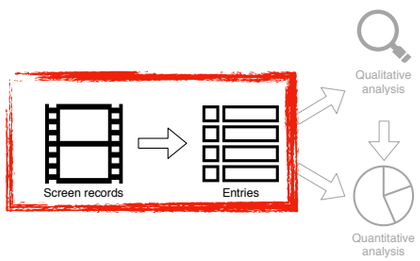
- Allow users to have multiple identifiers ➡ *account_alias* table is created
- Gather all the secondary identifiers of a *person* ➡ *login_alias* column

Experimental setup

- The architect of AppSI recorded his screen during 3 development sessions.
- A roadmap is used to keep track of the progress during the evolution.

Video #	Total time
1	1:55:30
2	1:19:17
3	0:52:41
Total	**04:07:28**





Decomposing the evolution

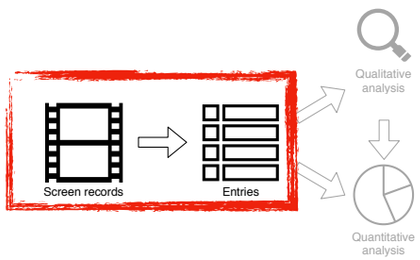
- Videos were transcribed as **entries** *(312 extracted)*

Time stamps **Description**

  - ’00:07:24,00:07:39,Go back on the trigger and remove it without error.’

- Entries are generalised as **actions** *(18 identified)*
  - ‘Observe patch’
  - ‘Execute DDL query from IDE’
  - ‘Inactivity’
  - ‘Other’

- Actions are grouped into **activities** *(7 identified)*
  - ‘Implement changes into queries’
  - ‘Execute queries in a transaction and commit’



Decomposing the evolution

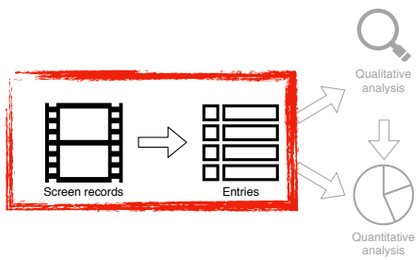
- Videos were transcribed as **entries** *(312 extracted)*

Time stamps **Description**

  - ’00:07:24,00:07:39,Go back on the trigger and remove it without error.’

- Entries are generalised as **actions** *(18 identified)*
  - ‘Observe patch’
  - ‘Execute DDL query from IDE’
  - ‘Inactivity’
  - ‘Other’

- Actions are grouped into **activities** *(7 identified)*
  - ‘Implement changes into queries’
  - ‘Execute queries in a transaction and commit’



Decomposing the evolution

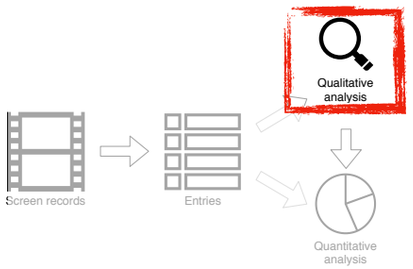
- Videos were transcribed as **entries** *(312 extracted)*

Time stamps **Description**

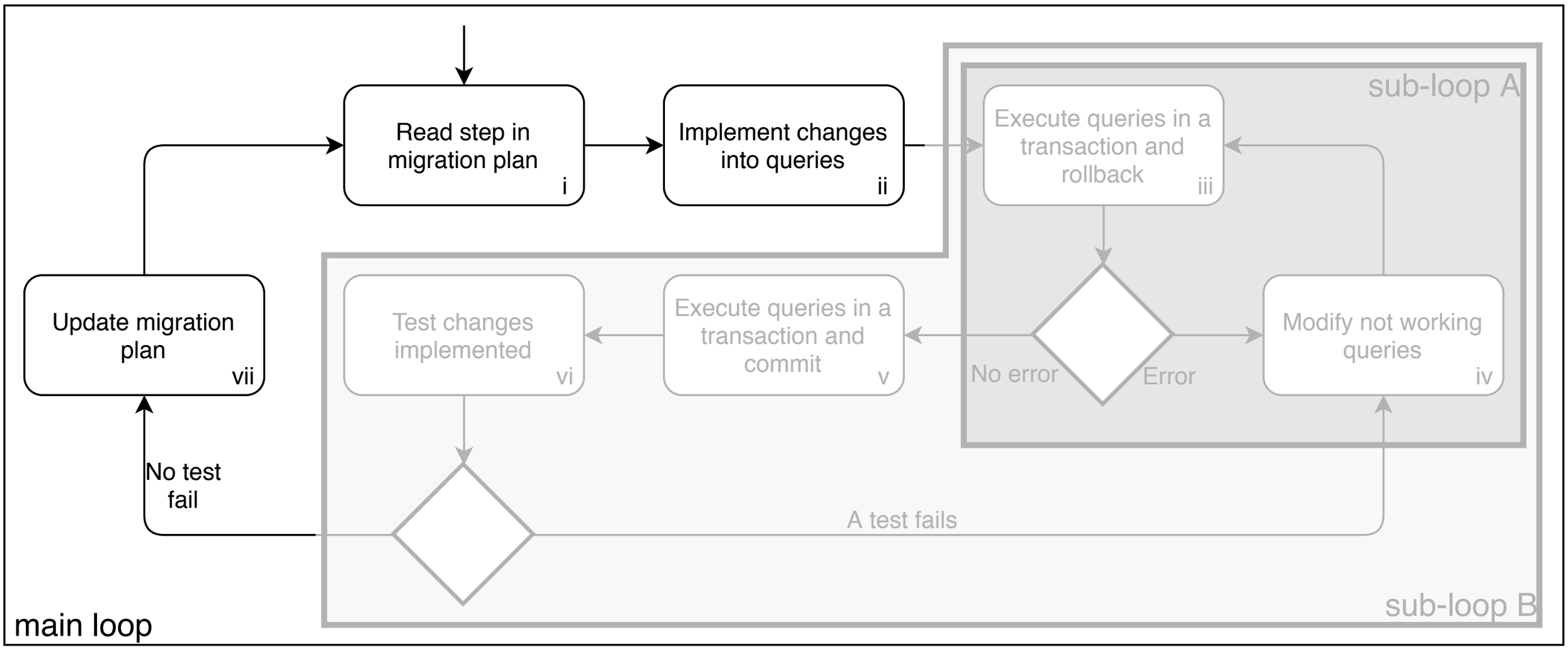
  - ’00:07:24,00:07:39,Go back on the trigger and remove it without error.’

- Entries are generalised as **actions** *(18 identified)*
  - ‘Observe patch’
  - ‘Execute DDL query from IDE’
  - ‘Inactivity’
  - ‘Other’

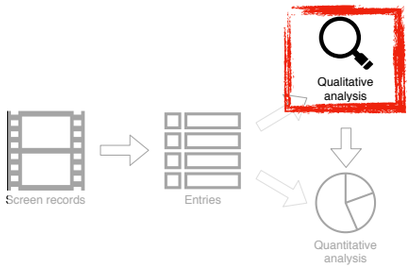
- Actions are grouped into **activities** *(7 identified)*
  - ‘Implement changes into queries’
  - ‘Execute queries in a transaction and commit’



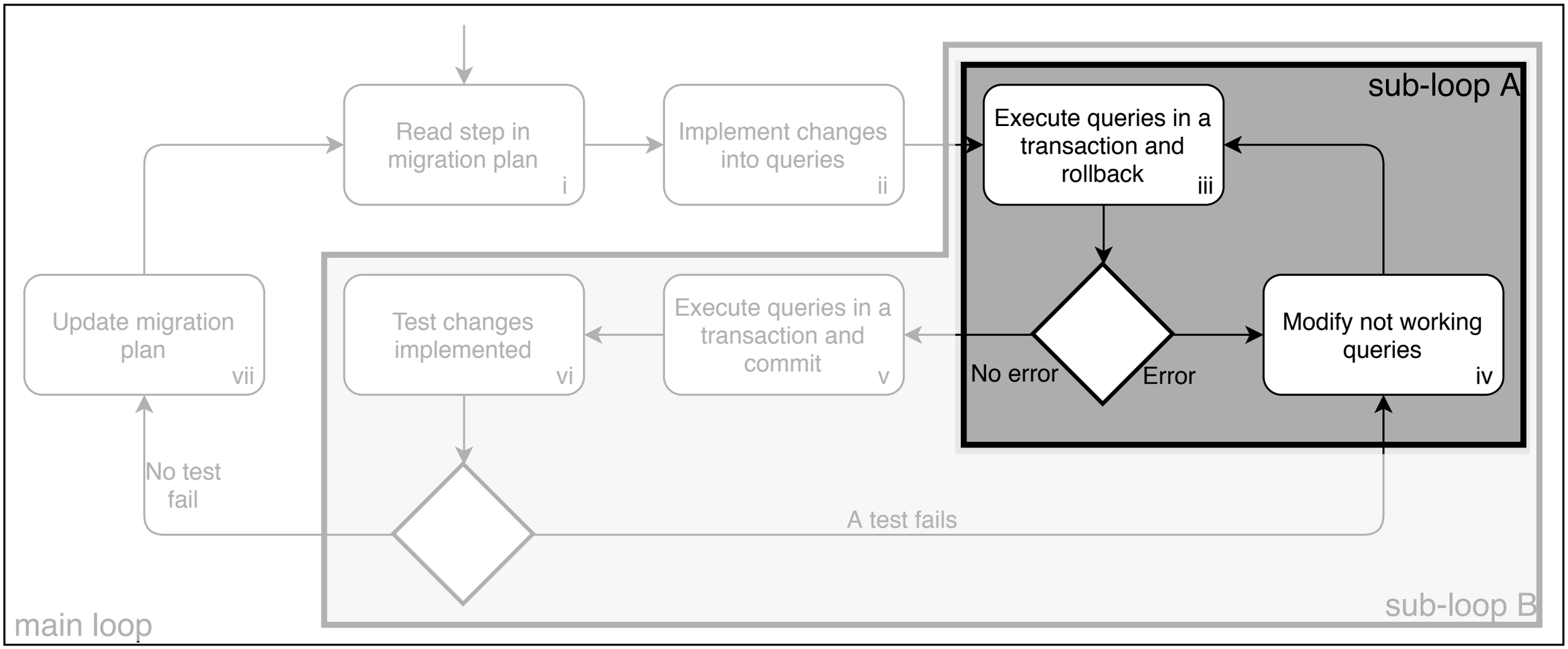
Process identified



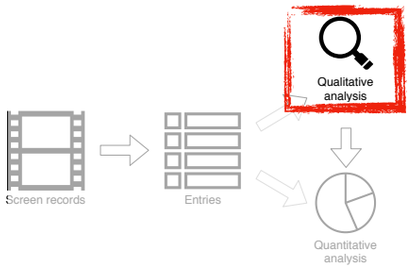
- **main loop**: complete implementation of a feature (might include multiple iterations on sub-loop A and/or sub-loop B).
- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.



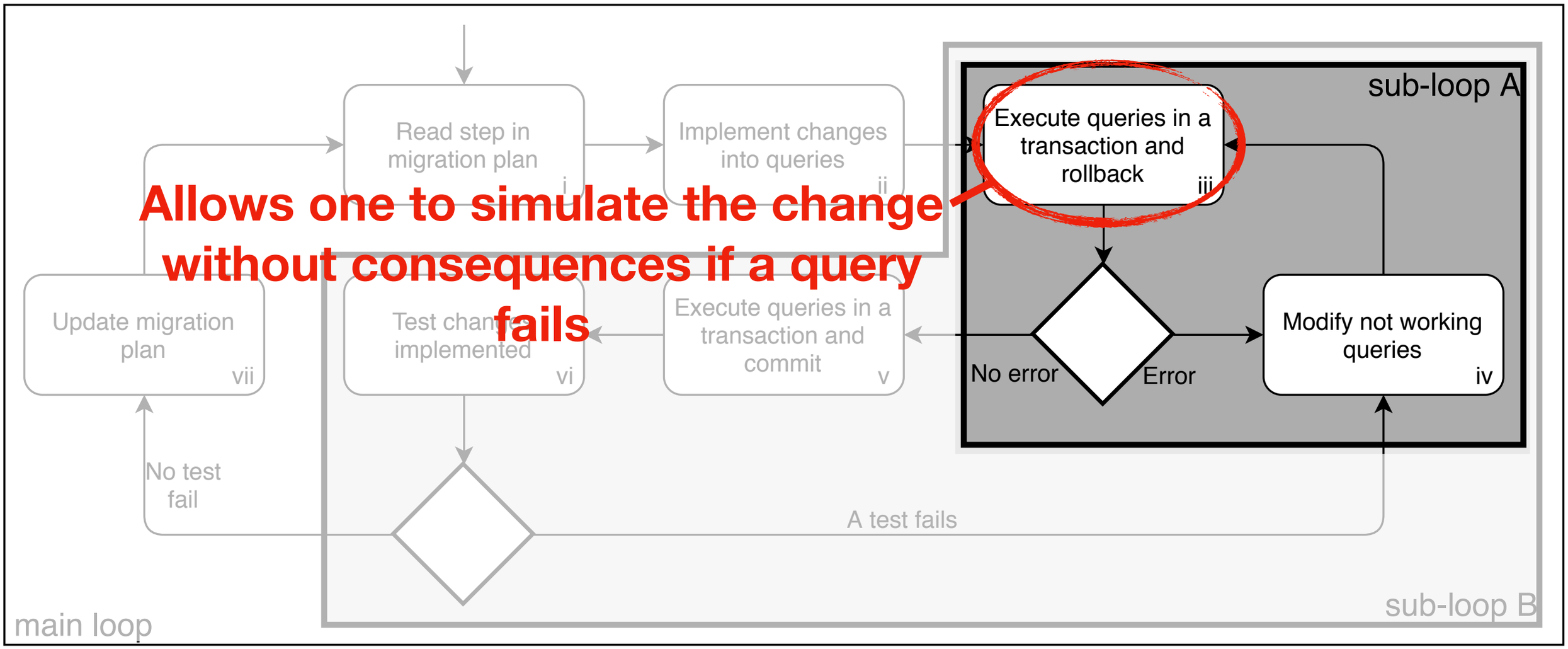
Process identified



- **main loop**: complete implementation of a feature (might include multiple iterations on sub-loop A and/or sub-loop B).
- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.

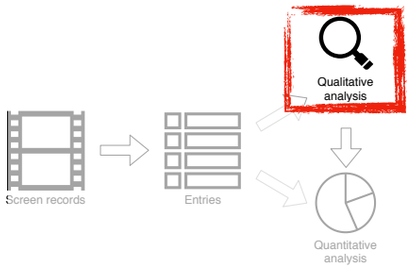


Process identified

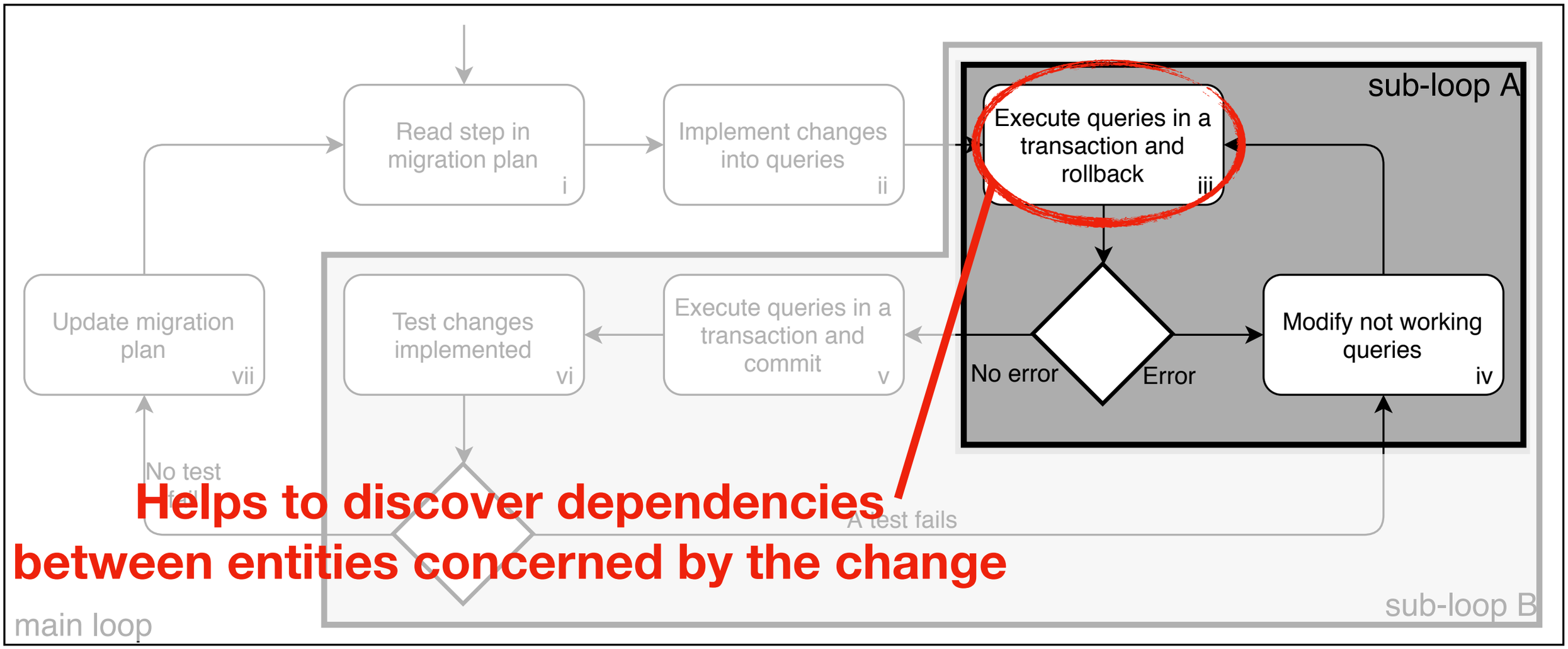


Allows one to simulate the change without consequences if a query fails

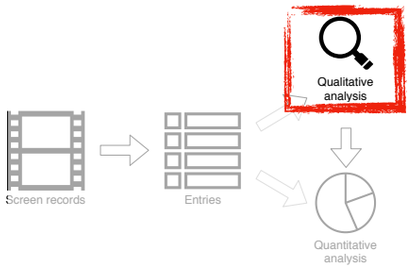
- **main loop**: complete implementation of a feature (might include multiple iterations on sub-loop A and/or sub-loop B).
- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.



Process identified



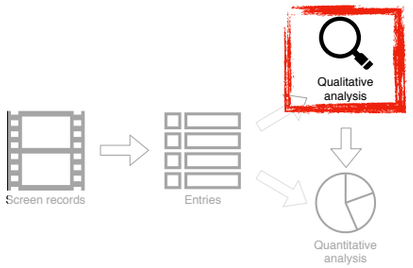
- **main loop**: complete implementation of a feature (might include multiple iterations on sub-loop A and/or sub-loop B).
- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.



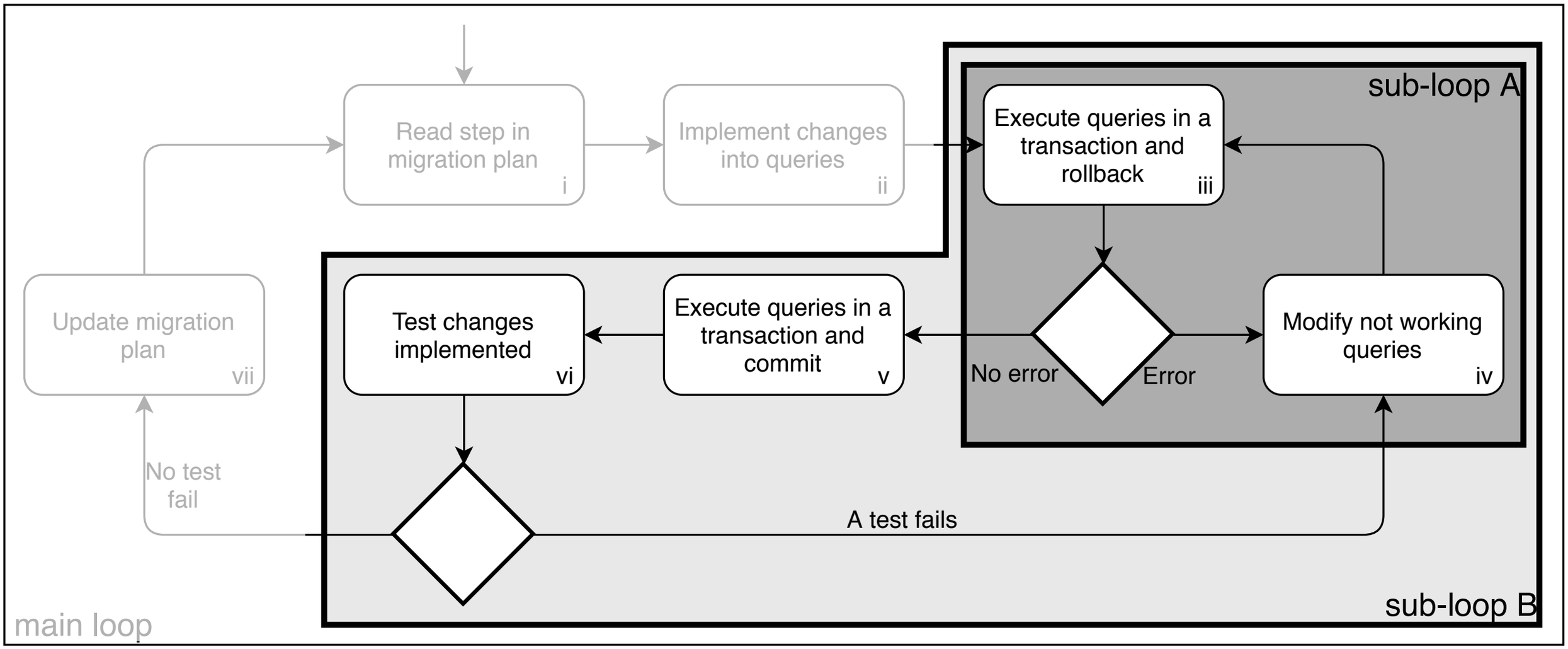
Process identified

Helps to handle the fact that RDBMS ensure consistency of the schema at any moment

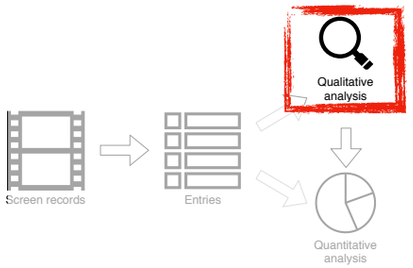
- **main loop**: complete implementation of a feature (might include multiple iterations on sub-loop A and/or sub-loop B).
- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.



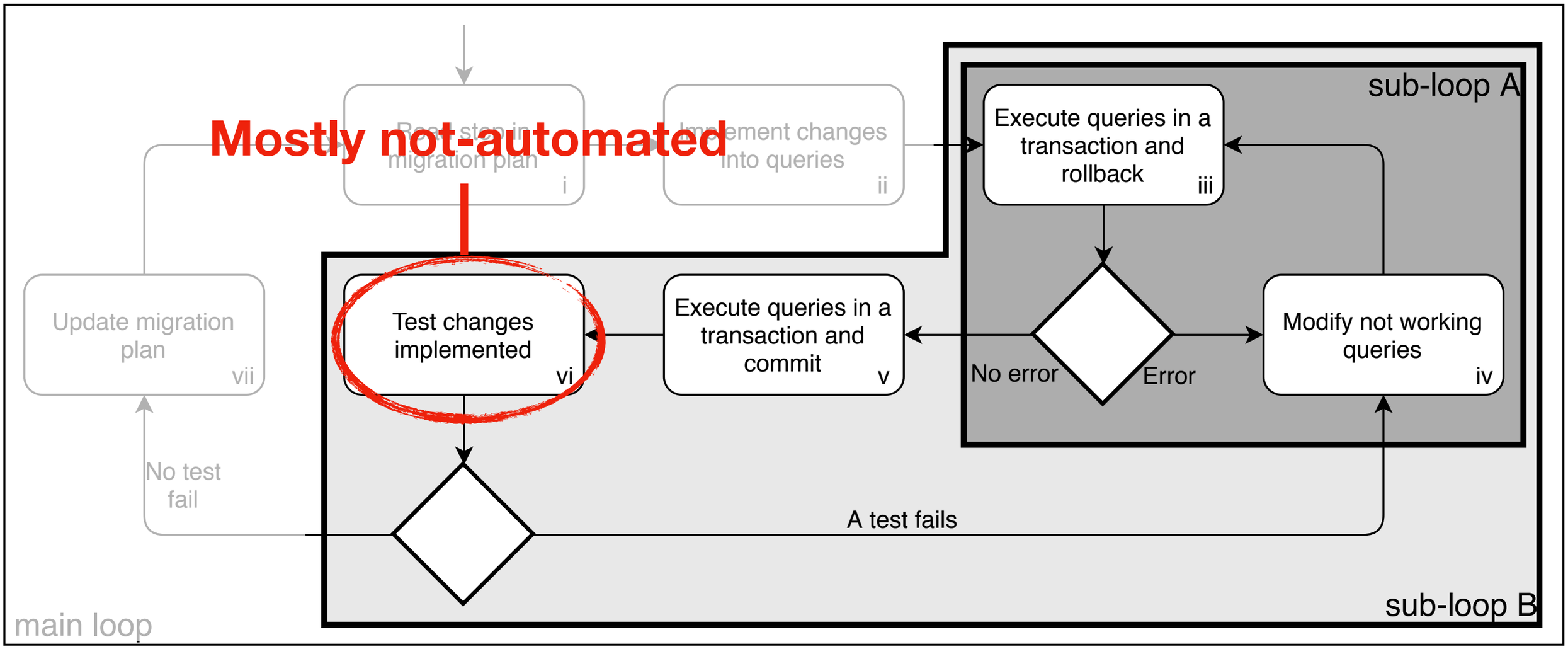
Process identified



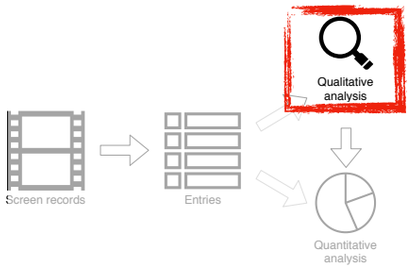
- **main loop**: complete implementation of a feature (might include multiple iterations on sub-loop A and/or sub-loop B).
- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.



Process identified

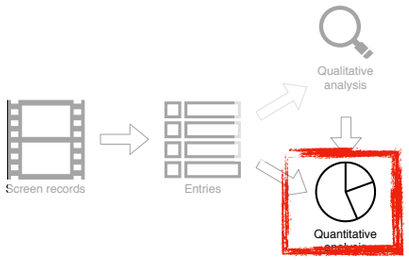


- **main loop**: complete implementation of a feature (might include multiple iterations on sub-loop A and/or sub-loop B).
- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.

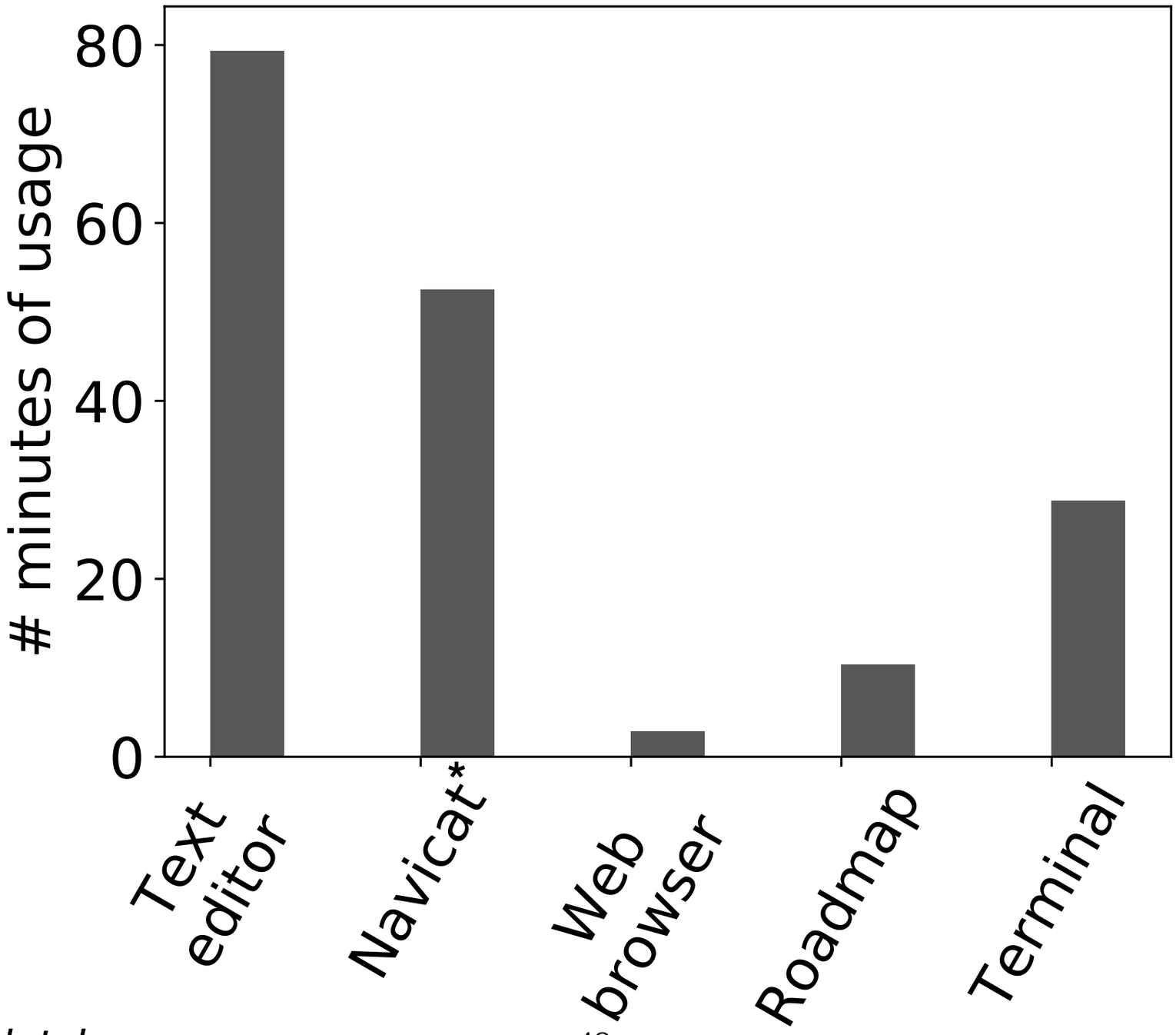


Process identified

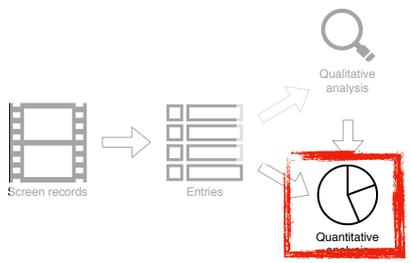
- **main loop**: complete implementation of a feature (might include multiple iterations on sub-loop A and/or sub-loop B).
- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.



Quantitative Analysis

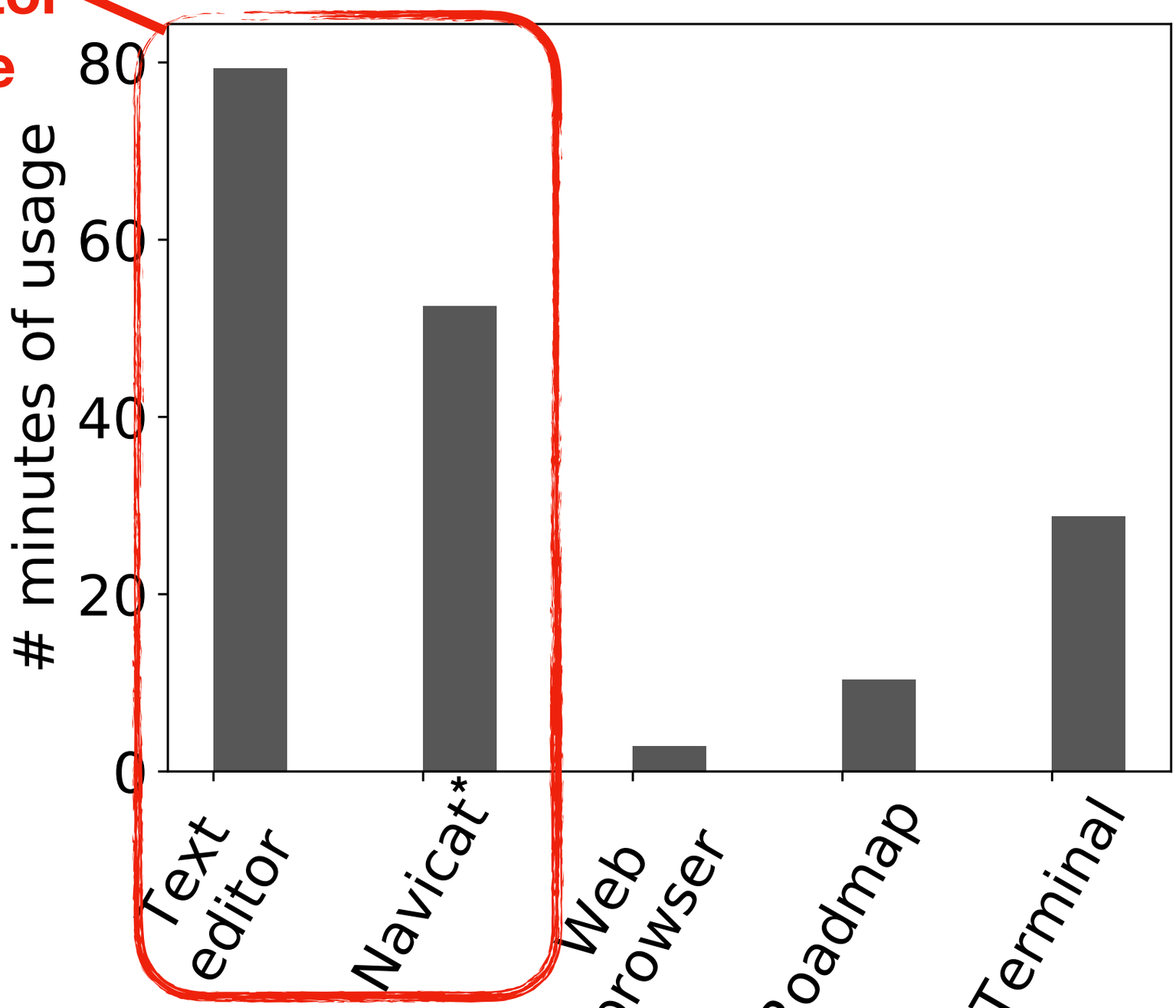


* *IDE for relational databases*

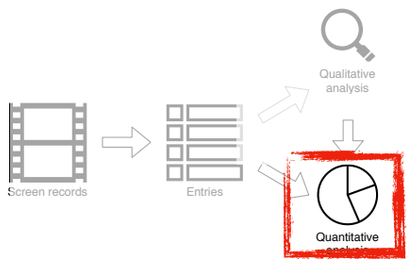


Quantitative Analysis

The text editor is used more time than the IDE

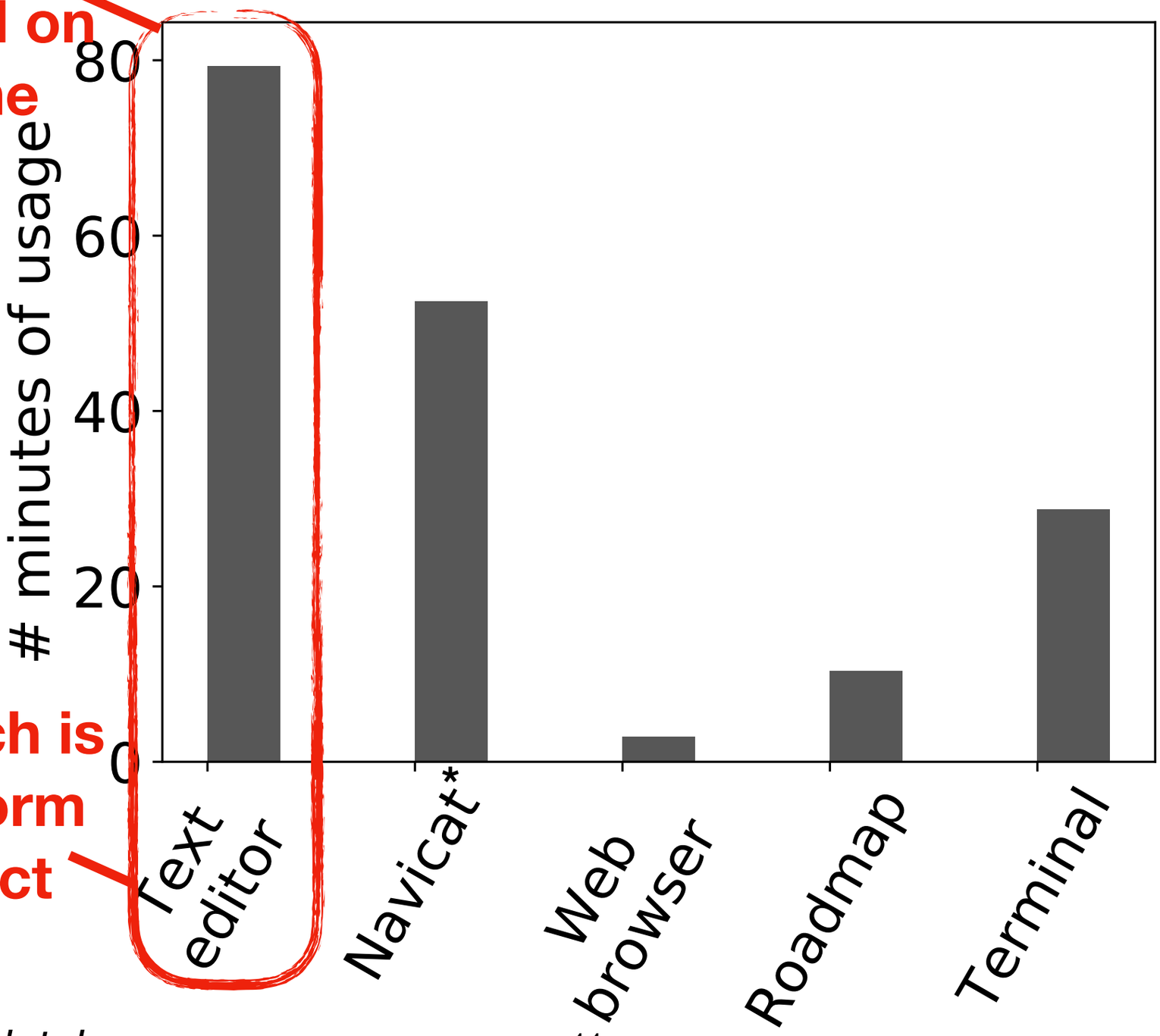


* *IDE for relational databases*



Quantitative Analysis

Reverse engineering is performed on a dump of the database



Textual search is used to perform change impact analysis

* *IDE for relational databases*

Problems identified

Problems

1. **Relational databases are hard to understand.**

    - RDBMS meta-data are complex to query.

    - Some meta-data are missing.

2. **Relational databases are hard to evolve.**

    - Dependencies between a database and a program using it are usually implicit.

    - The database schema can not be in an inconsistent state at any moment.

Problems

1. **Relational databases are hard to understand.**

    - RDBMS meta-data are complex to query.

    - Some meta-data are missing.

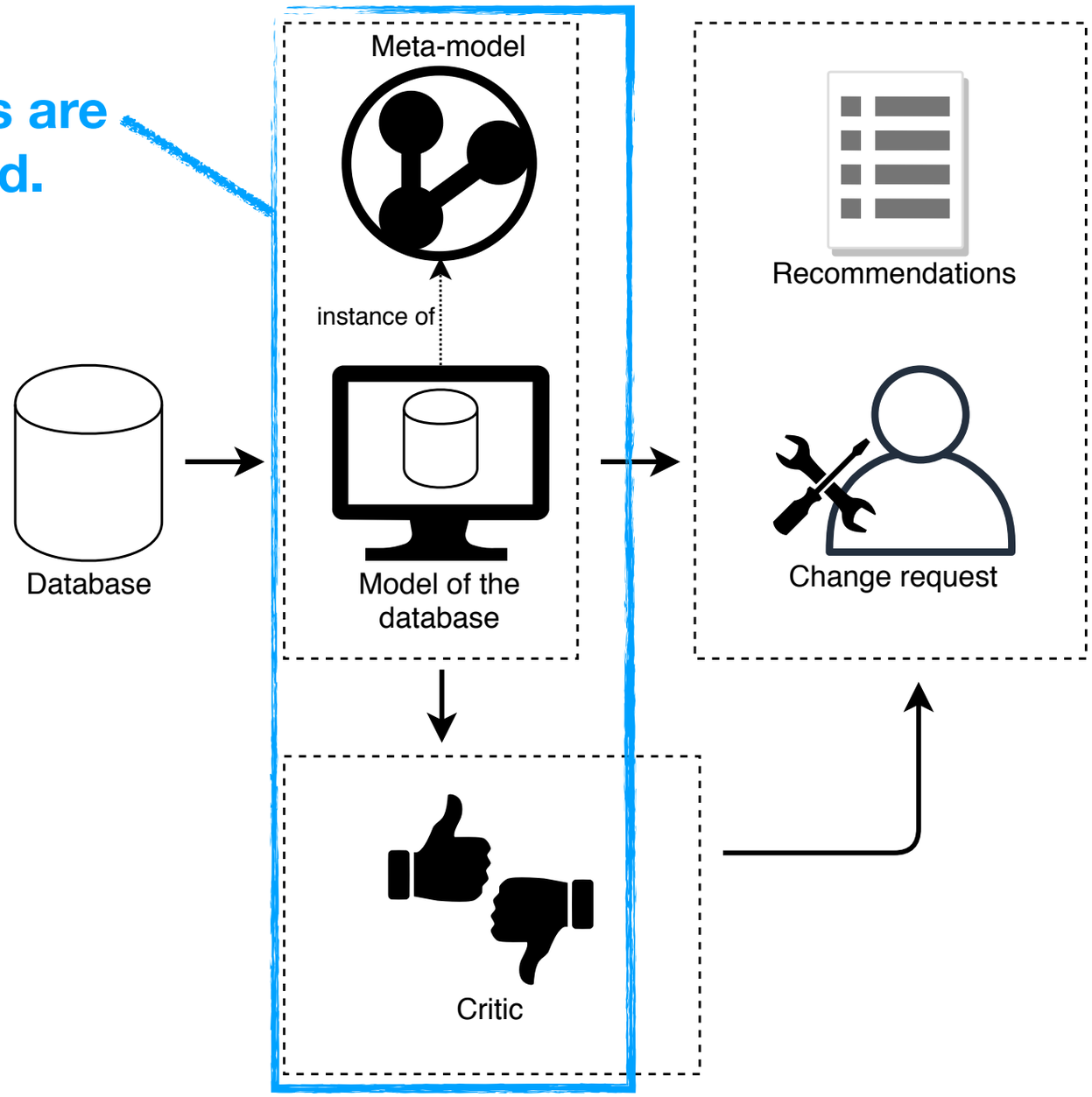
2. **Relational databases are hard to evolve.**

    - Dependencies between a database and a program using it are usually implicit.

    - The database schema can not be in an inconsistent state at any moment.

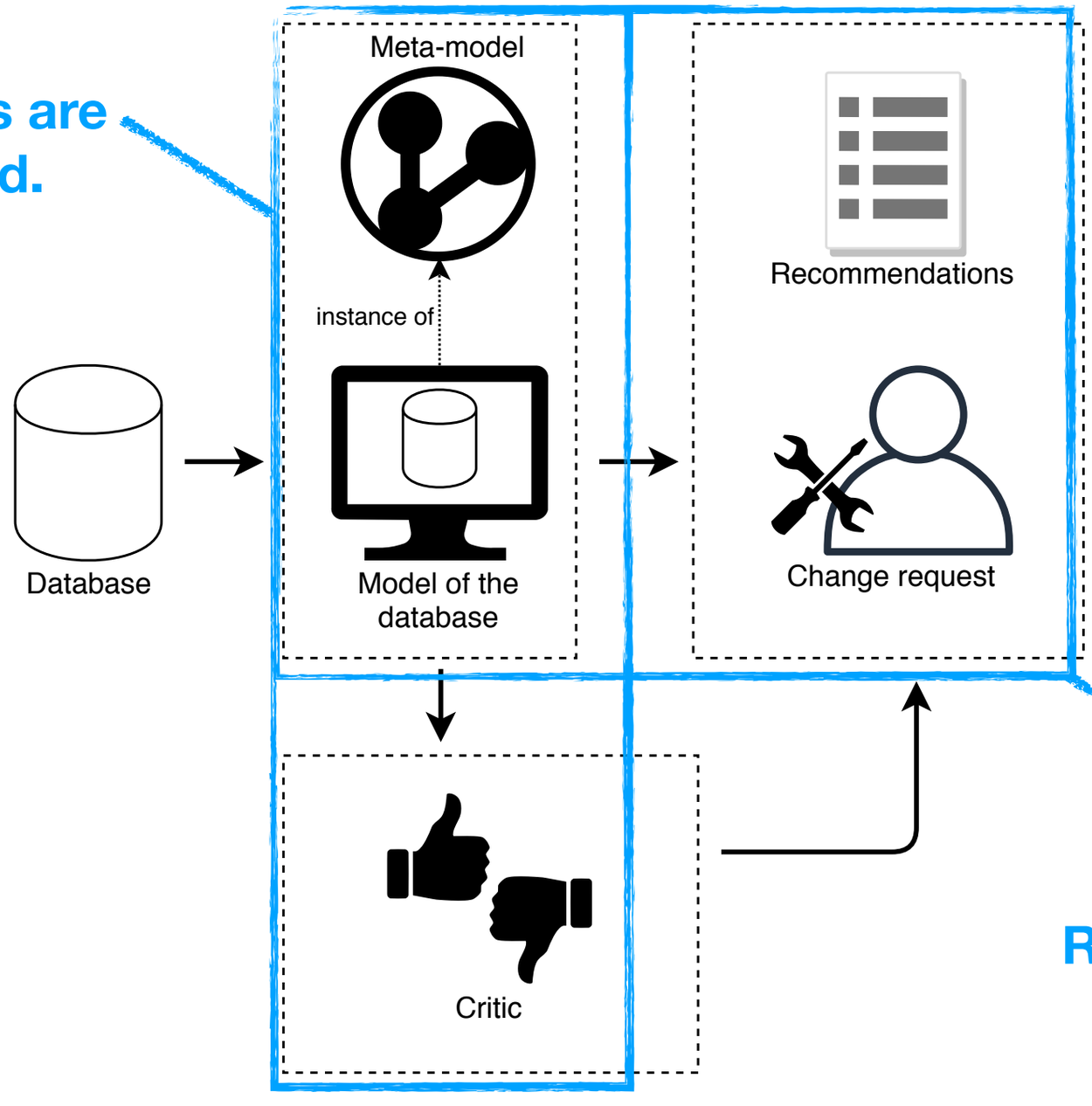
Our approach: Software Engineering for Relational Databases

Problem 1:
Relational databases are hard to understand.



Our approach: Software Engineering for Relational Databases

Problem 1:
Relational databases are hard to understand.

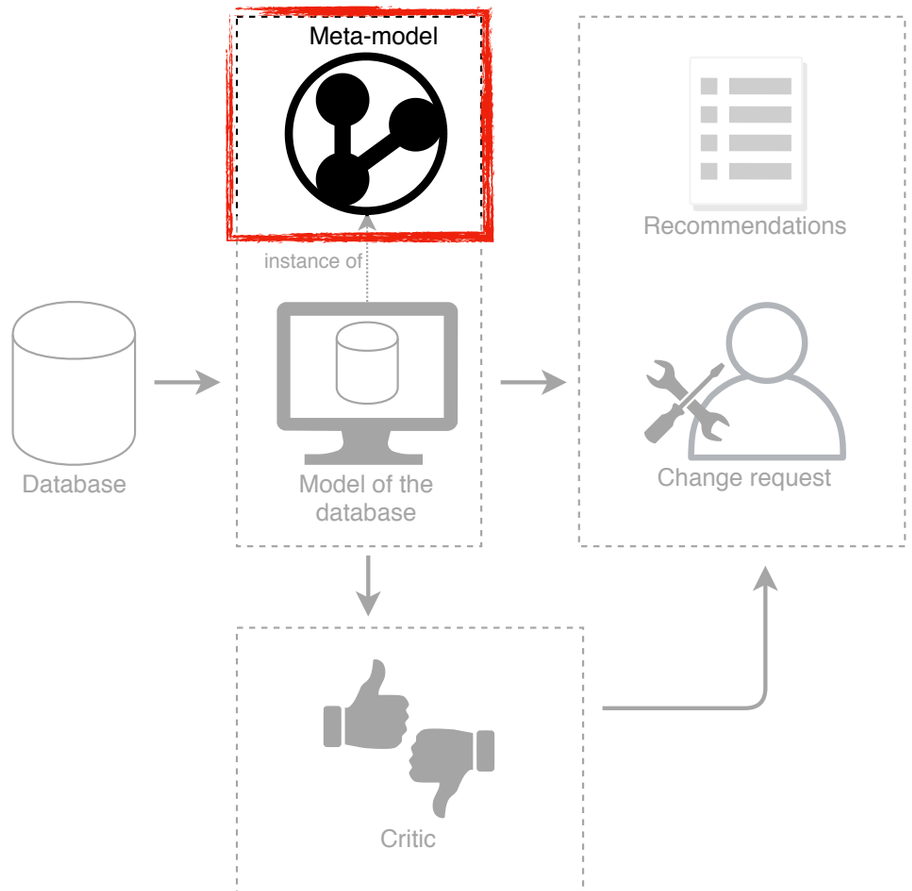


Problem 2:
Relational databases are hard to evolve.

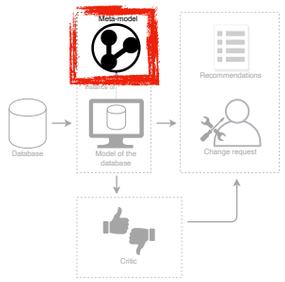
Roadmap

- Context & State of the Art
- Motivation
- **A Behavior-Aware Meta-Model for Relational Databases**
- Identifying Quality Issues in Relational Databases
- Recommendations for Evolving Relational Databases
- Conclusion

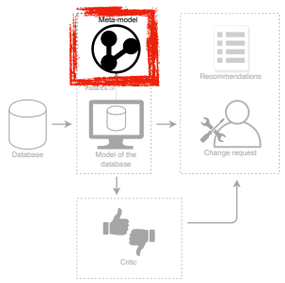
A Behavior-Aware Meta-Model for Relational Databases



Recommendations for Evolving Relational Databases. CAISE’20

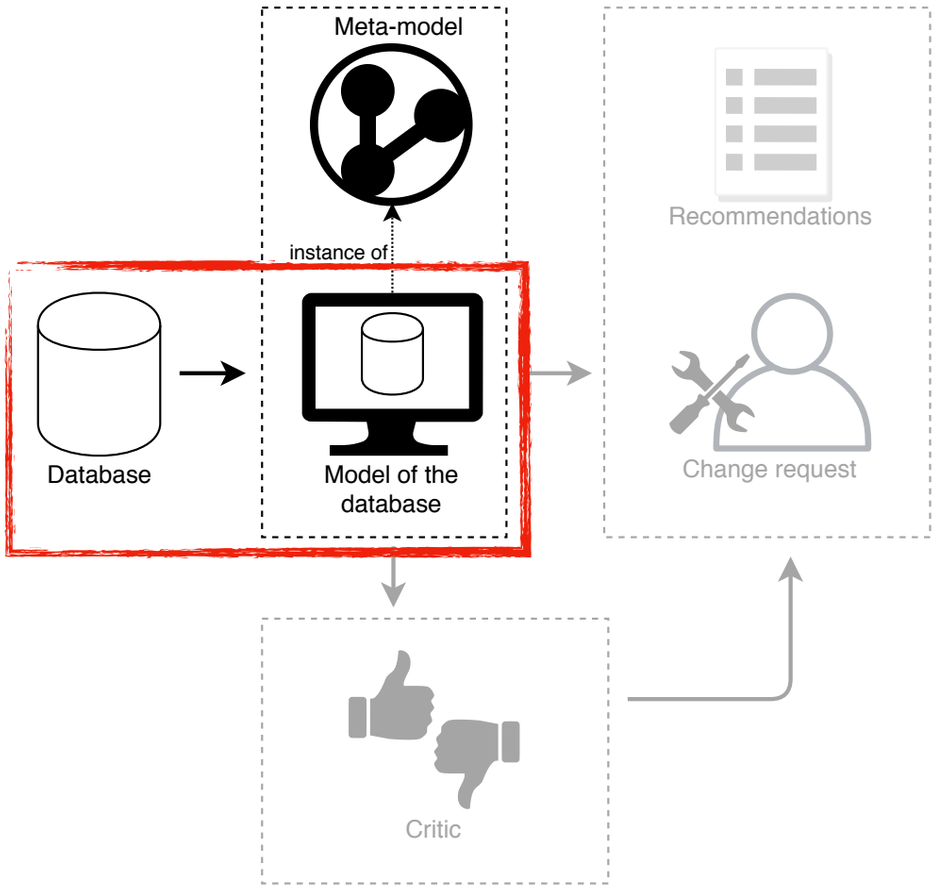


Meta-model: Structural entities



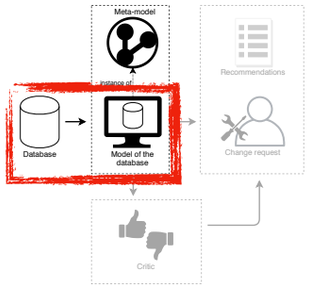
Meta-model: Behavioral entities

Instantiating the Meta-Model

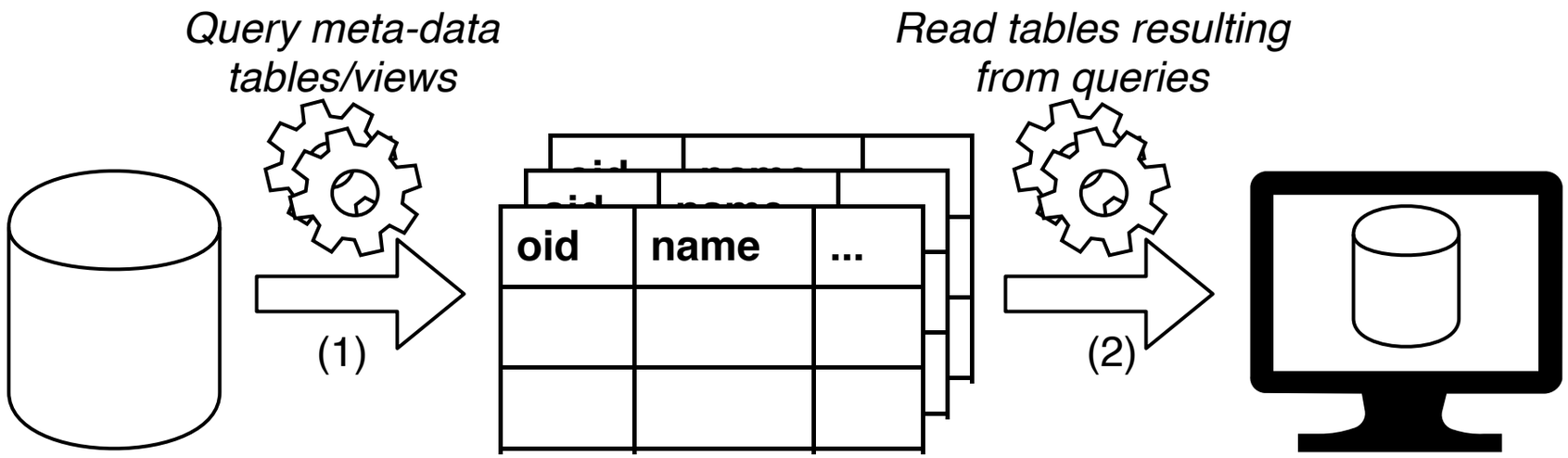


Evaluation of 3 approaches according to 3 criteria:

1. Completeness
2. Sensibility to RDBMS evolution
3. Complexity to migrate to another RDBMS

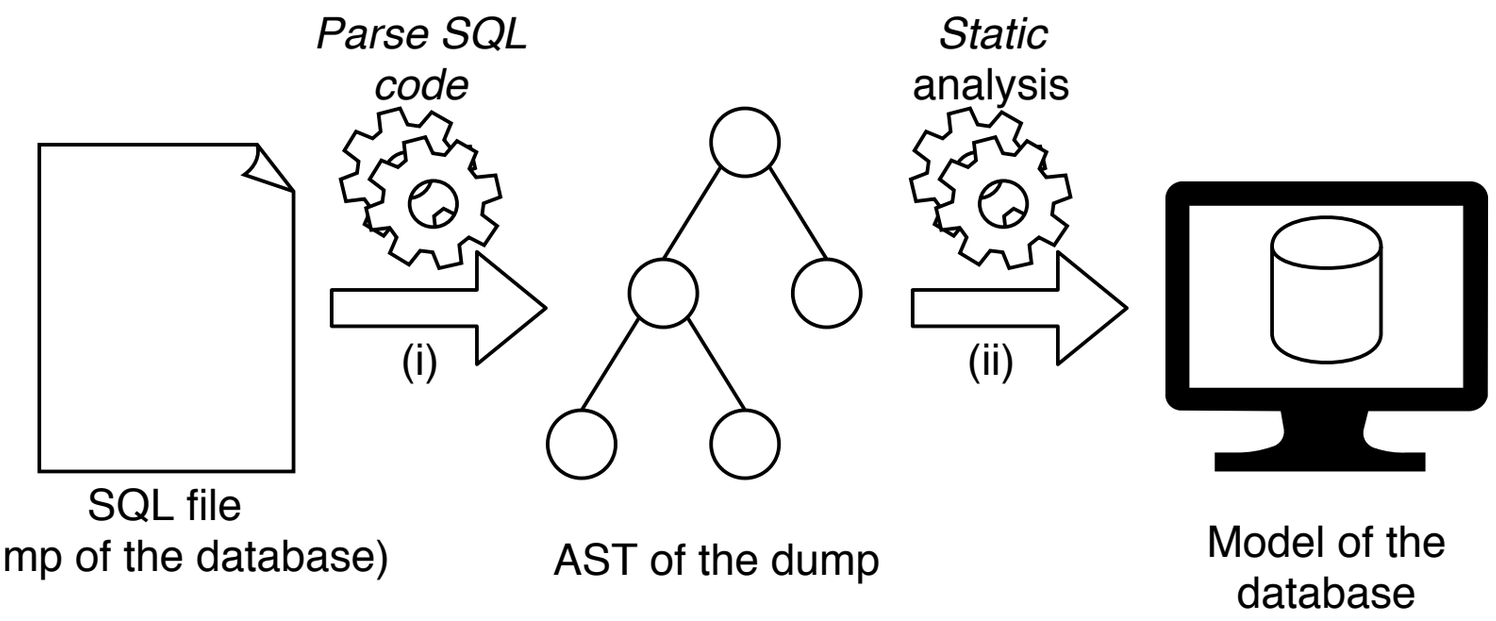


Meta-Data Analysis



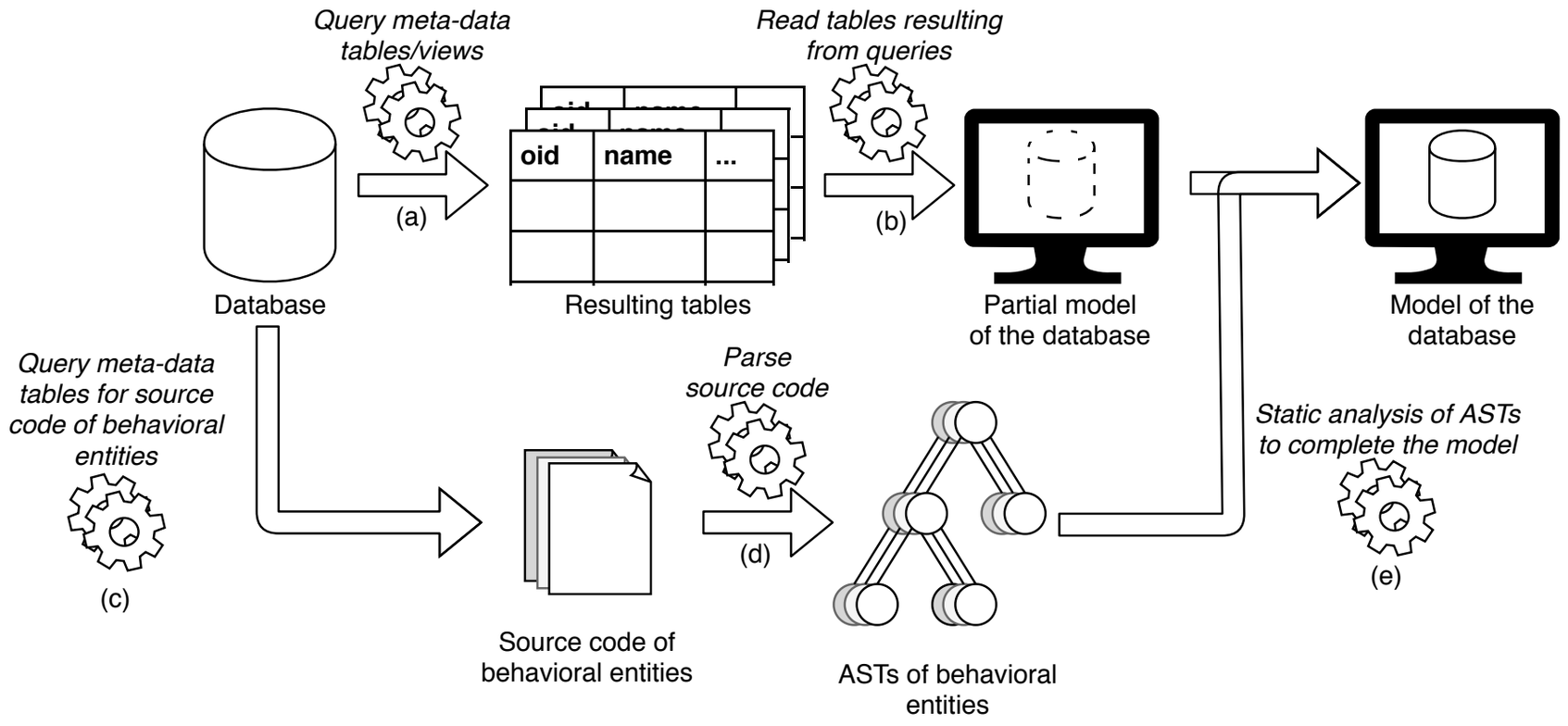
Approach	Completeness	Sensitivity to RDBMS evolution	Complexity to migrate to another RDBMS
Meta-data Analysis	*incomplete*	*low*	*simple*

Dump Analysis



Approach	Completeness	Sensitivity to RDBMS evolution	Complexity to migrate to another RDBMS
Meta-data Analysis	*incomplete*	*low*	*simple*
Dump Analysis	*complete*	*high*	*complex*

Hybrid Approach



Approach	Completeness	Sensitivity to RDBMS evolution	Complexity to migrate to another RDBMS
Meta-data Analysis	*incomplete*	*low*	*simple*
Dump Analysis	*complete*	*high*	*complex*
Hybrid Approach	*complete*	*limited* Tradeoff	*moderate* Tradeoff

Meta-model Evaluation

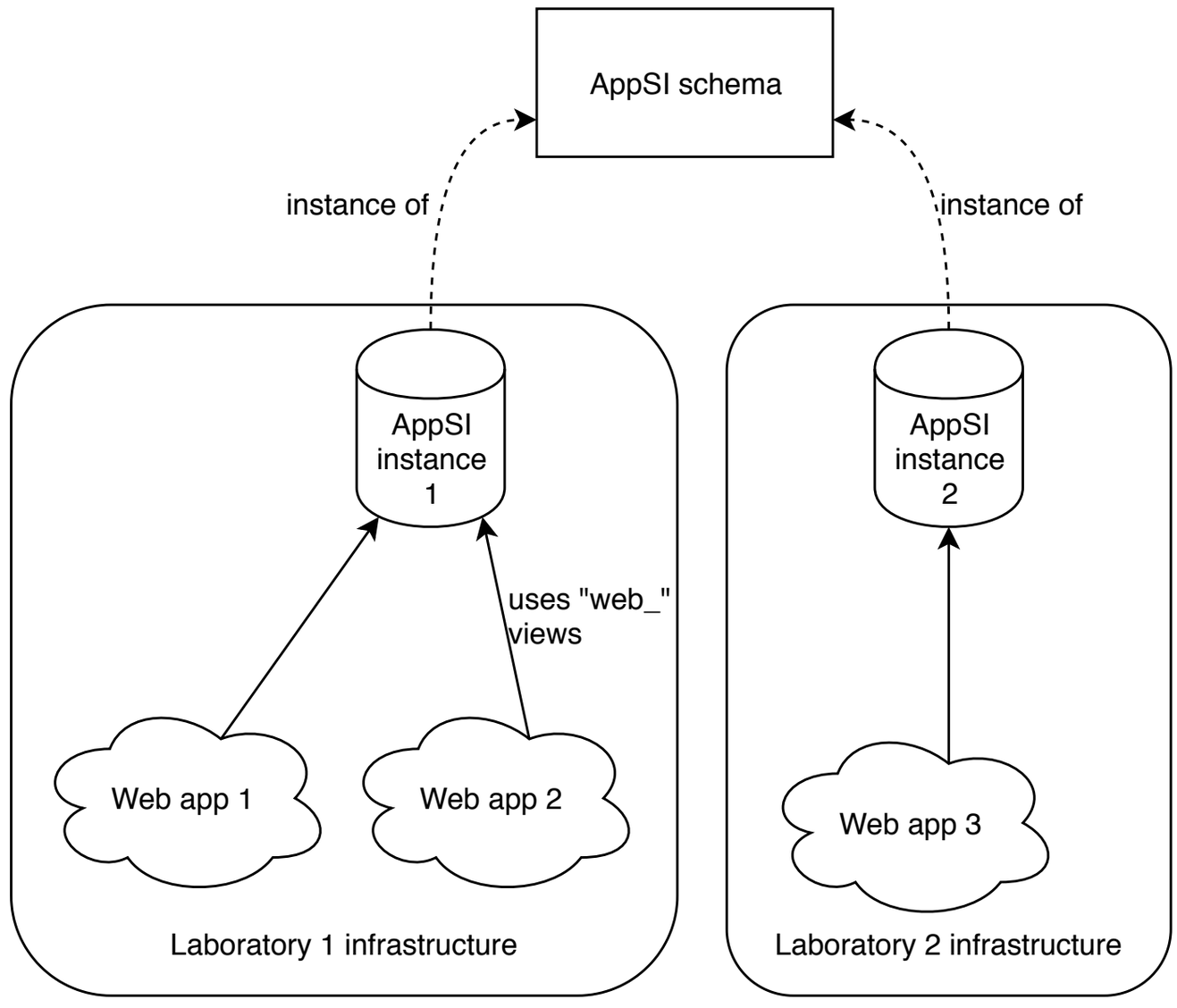
Goal: illustrate that the problem « Relational databases are hard to understand » is addressed by the meta-model.

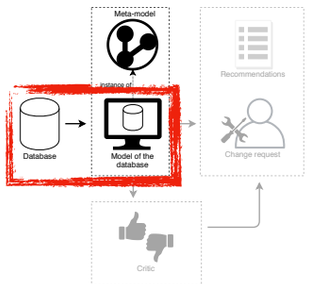
- Case study 1: Queries to support (re)modularization
- Case study 2: Retrieving Features Implemented by a Database

Case study 1: Queries to support (re)modularization

Context:

- Laboratory 1 & 2 are running 2 different instances of AppSI schema
- Multiple applications can use an instance.
- « web views » are only used by Web app 2

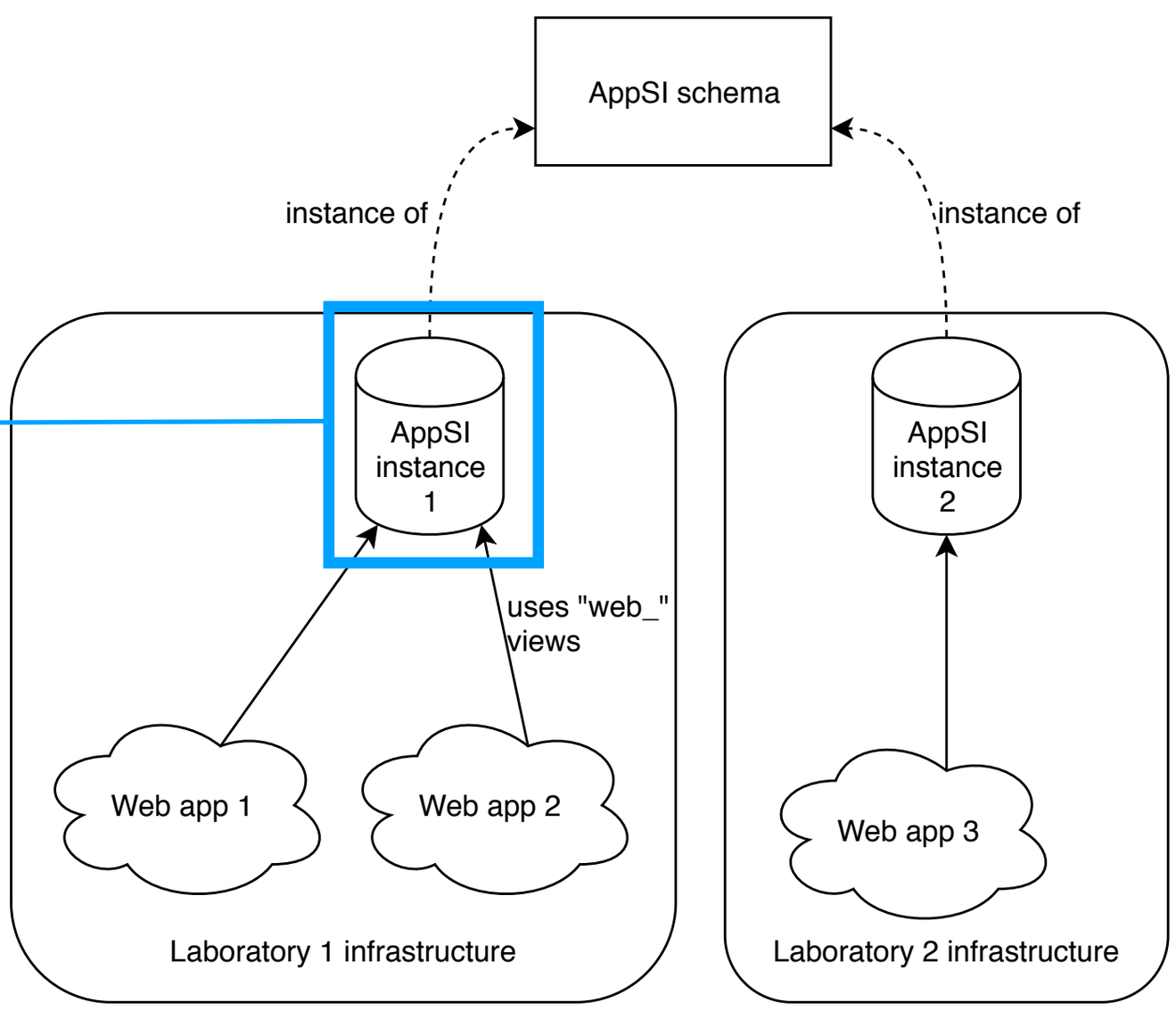


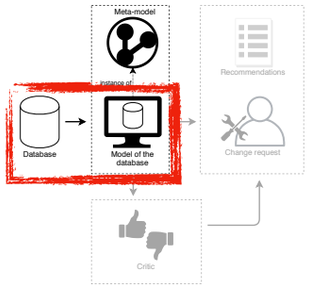


Case study 1: Queries to support (re)modularization

Problem:

- Web views should only be deployed on instances of AppSI that needs them.
- The current DBA of AppSI is not the person who created these views.



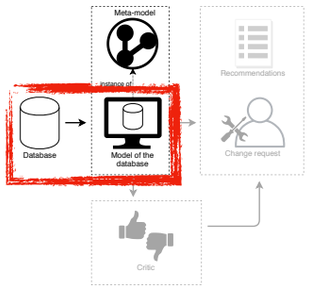


Case study 1: Queries to support (re)modularization

Methods:

1. Extract dependency relationships between web views and the rest of the database.

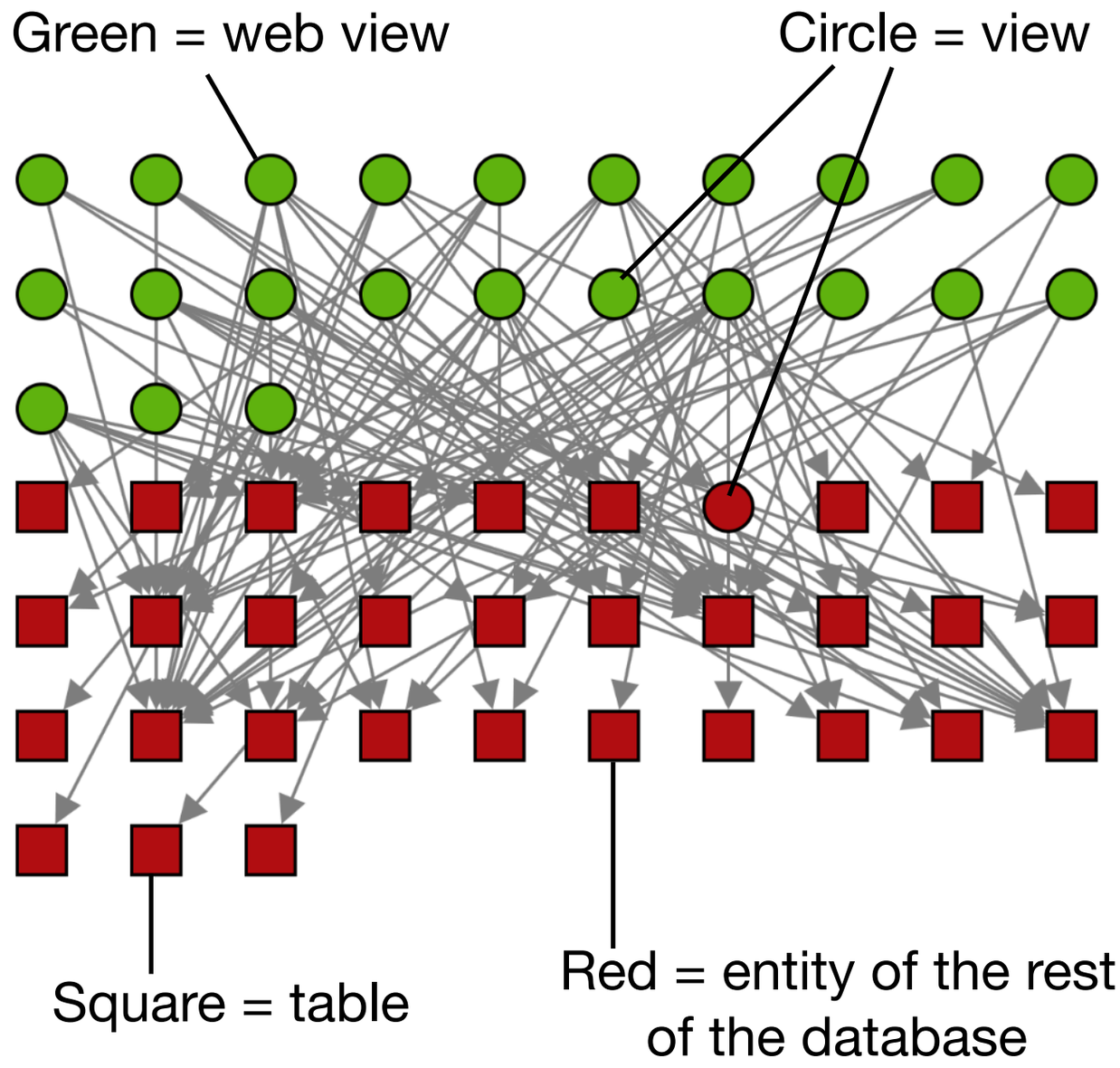
2. Analyze dependencies to take a decision on how to move web views to a separated namespace.

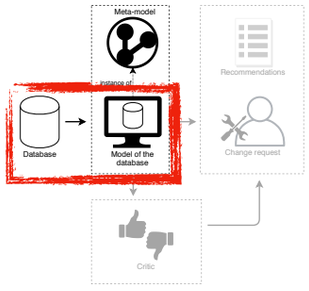


Case study 1: Queries to support (re)modularization

Results:

- **No entity** of the database **depends on a web view**.
- **Entities** of AppSI **web views depend on** are revealed.
- **Web views were moved** to a separated namespace without problem.

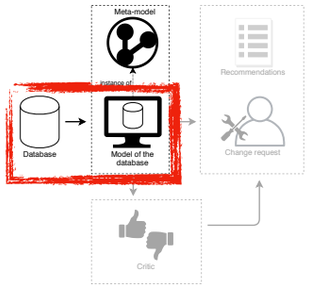




Case study 2: Retrieving Features Implemented by a Database

Context:

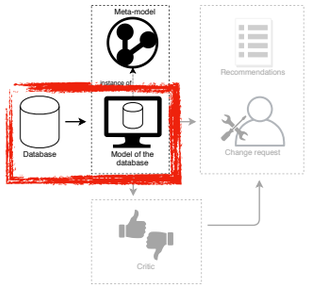
- AppSI evolves to fulfil new requirements of the laboratory.
- These requirements evolve with time depending on administrative processes.



Case study 2: Retrieving Features Implemented by a Database

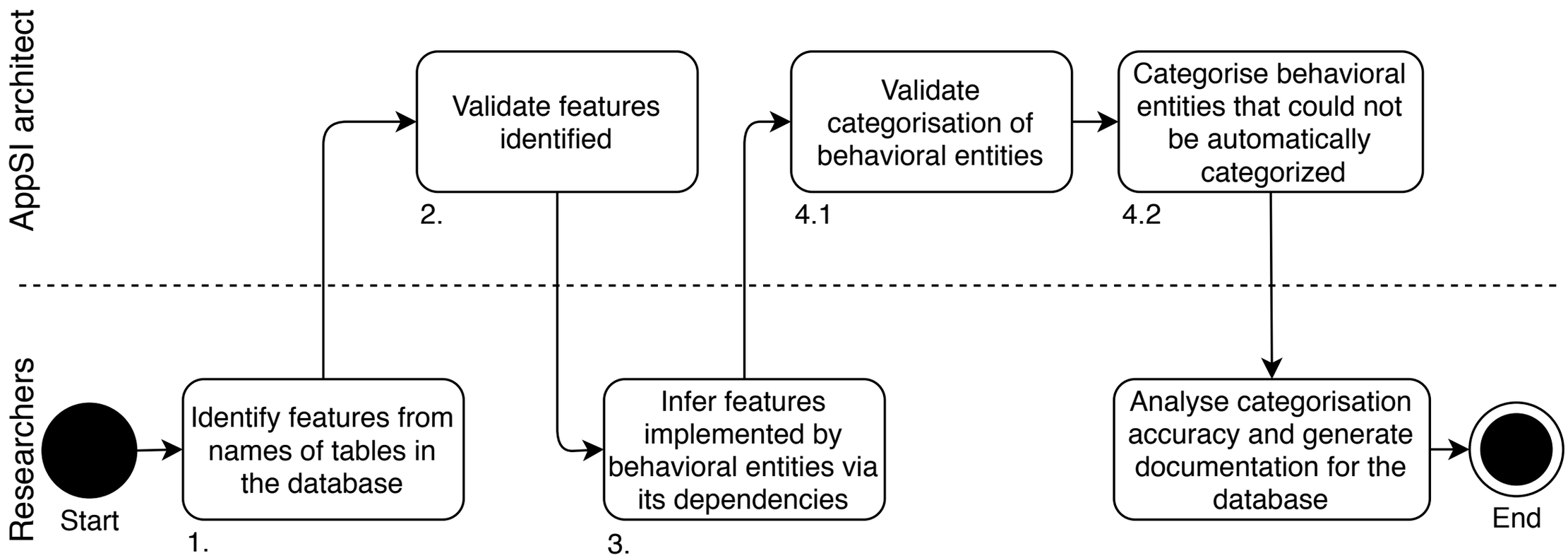
Problem:

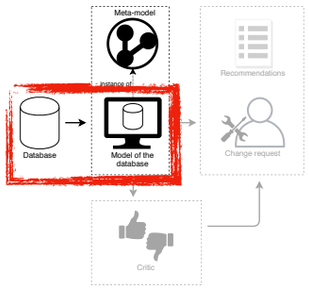
- Features implemented by AppSI emerged from successive evolutions of the database.
- Not clearly defined nor documented.
- Getting an overview of the database by reading the source code of its schema becomes harder and harder with time.



Case study 2: Retrieving Features Implemented by a Database

Methods (in a nutshell):





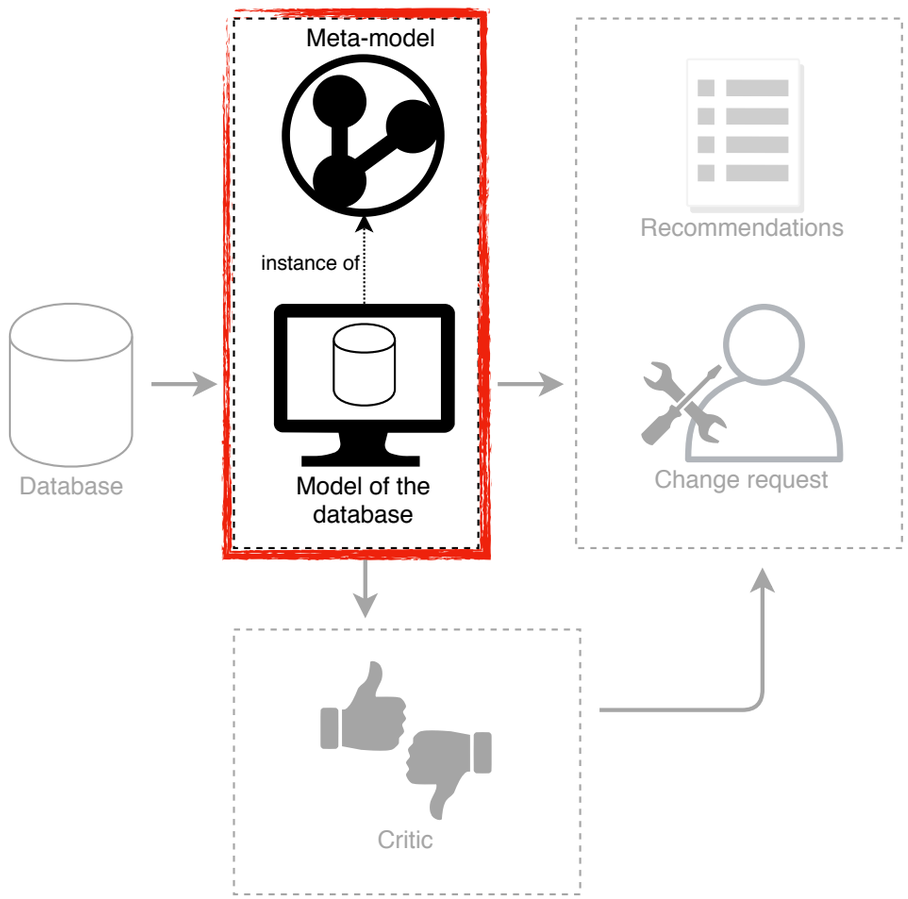
Case study 2: Retrieving Features Implemented by a Database

Results:

- Identified **11 feature groups** validated by the architect.
- **97%** of behavioral entities are **correctly assigned** to their feature group based on their dependencies.
- **Valuable documentation** for future maintainers of AppSI.

Feature group	# Tables	# View	# Stored Proc.	# Trigger	# Trigger Stored Proc.
Members management	31	24	30	13	18
Buildings management	8	0	1	0	0
Mailing list	4	1	0	0	0
PhD and HDR management	12	17	6	11	9
Public website	6	8	0	2	3
Structure of laboratory	24	24	10	4	7
Access management	1	0	0	1	0
System tables	16	0	1	0	0
Profile	2	0	0	0	0
Scientific production	4	1	0	0	0
Utilities	0	0	11	0	0

Summary



- Meta-Model **representing** both **structural** and **behavioral** entities.

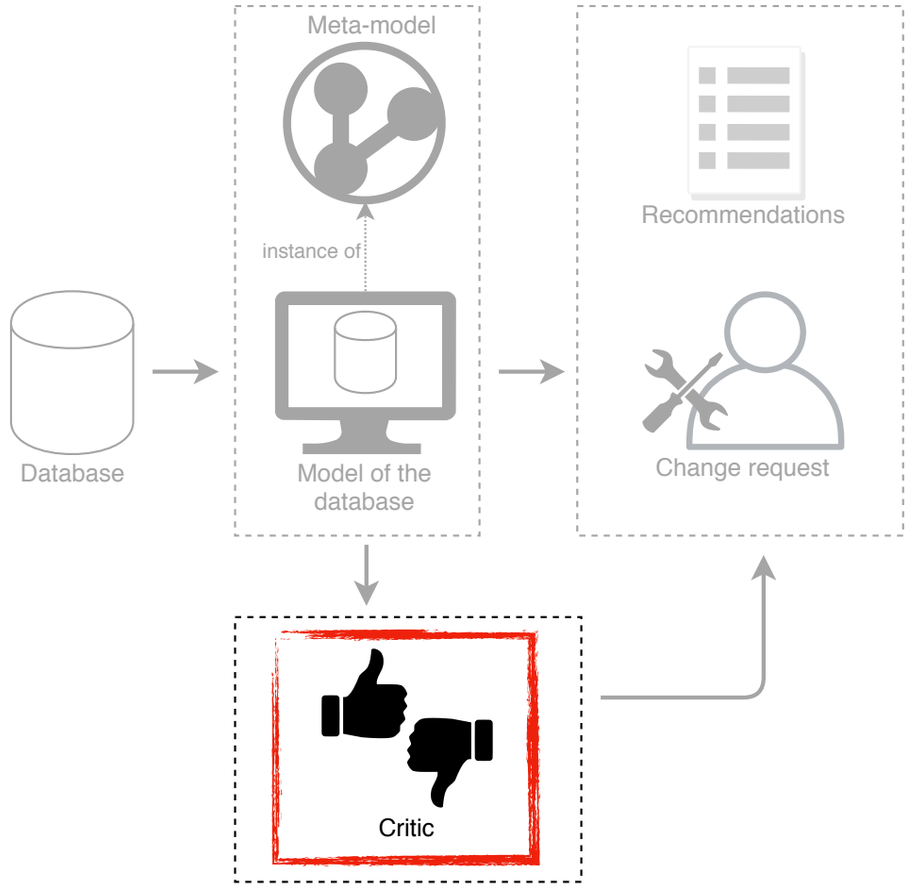
- **Evaluation** of approaches to **instantiate the meta-model** and proposition of an **hybrid approach**.

- Case studies illustrating that **our meta-model addresses the problem « Relational databases are hard to understand »**.

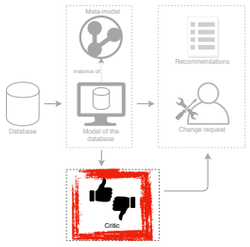
Roadmap

- Context & State of the Art
- Motivation
- A Behavior-Aware Meta-Model for Relational Databases
- ▸ **Identifying Quality Issues in Relational Databases**
- Recommendations for Evolving Relational Databases
- Conclusion

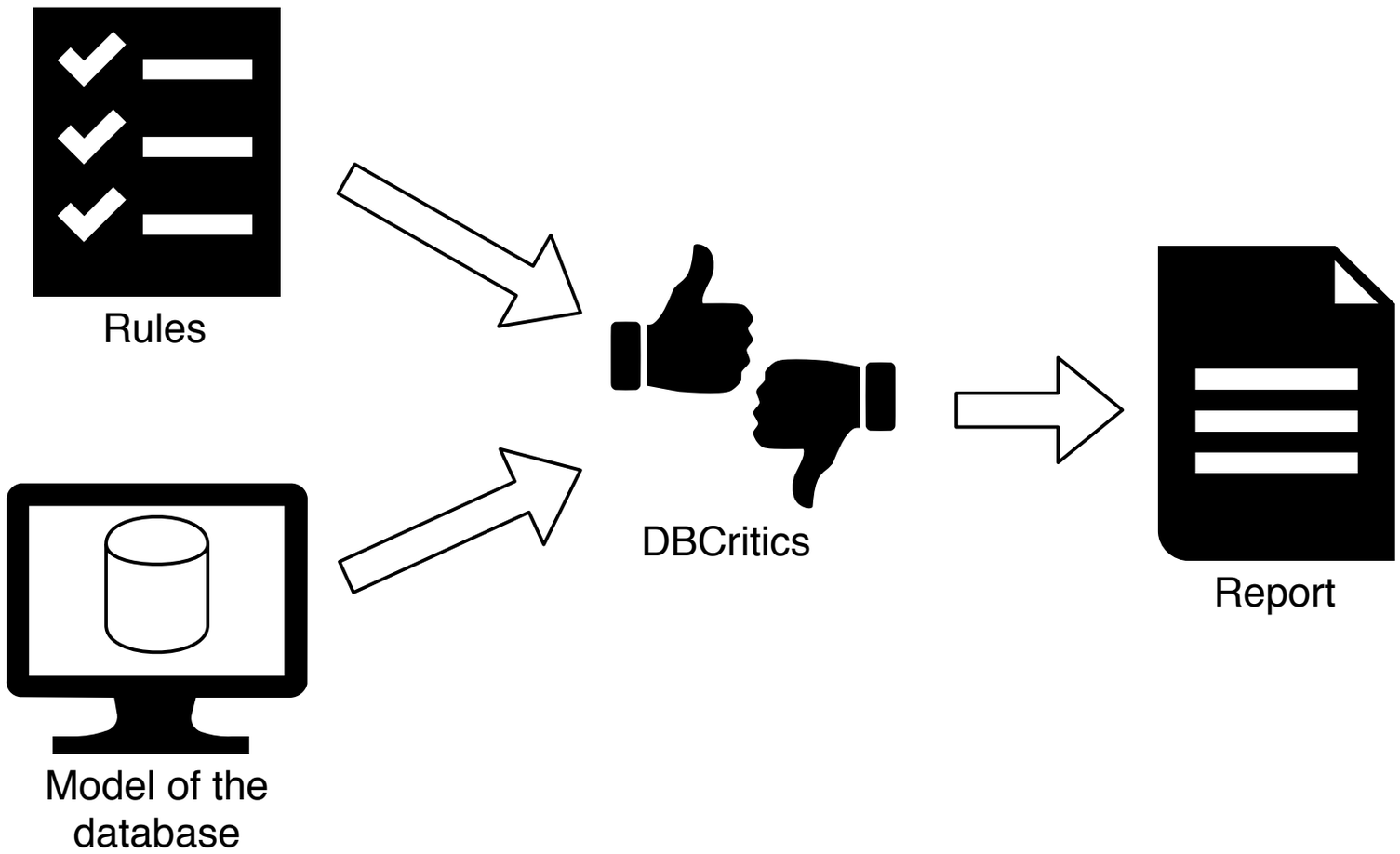
Identifying Quality Issues in Relational Databases



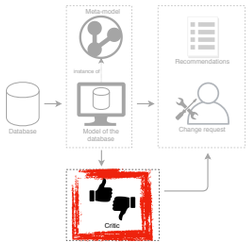
CodeCritics Applied to Database Schema: Challenges and First Results. SANER’17



DBCritics



Apply traditional Software Quality Analysis methods to database schemas



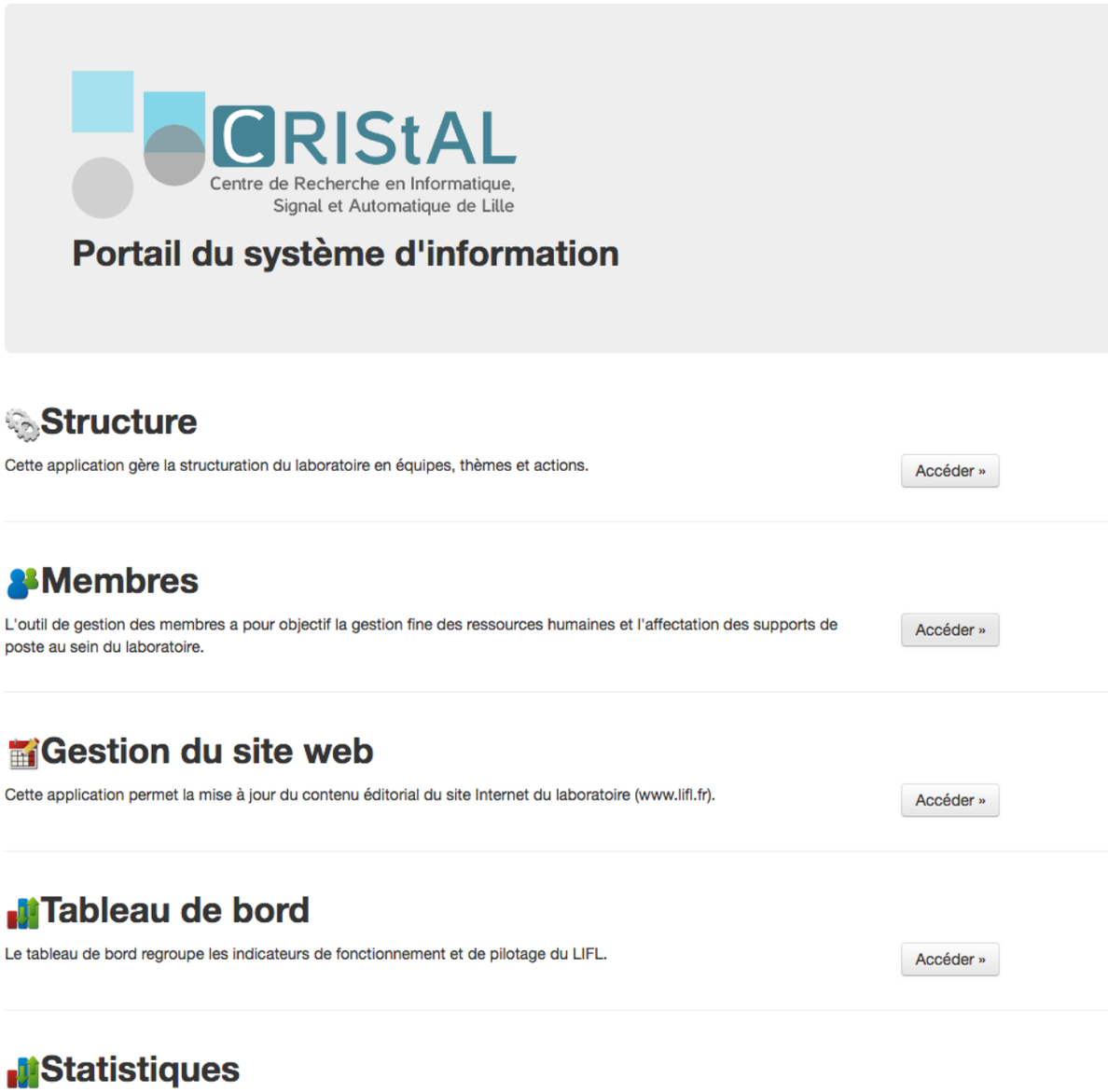
Evaluation

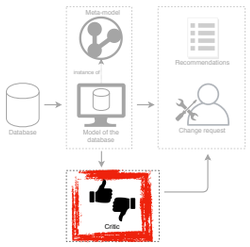
- WikiMedia: 25 versions analysed

Open-source database

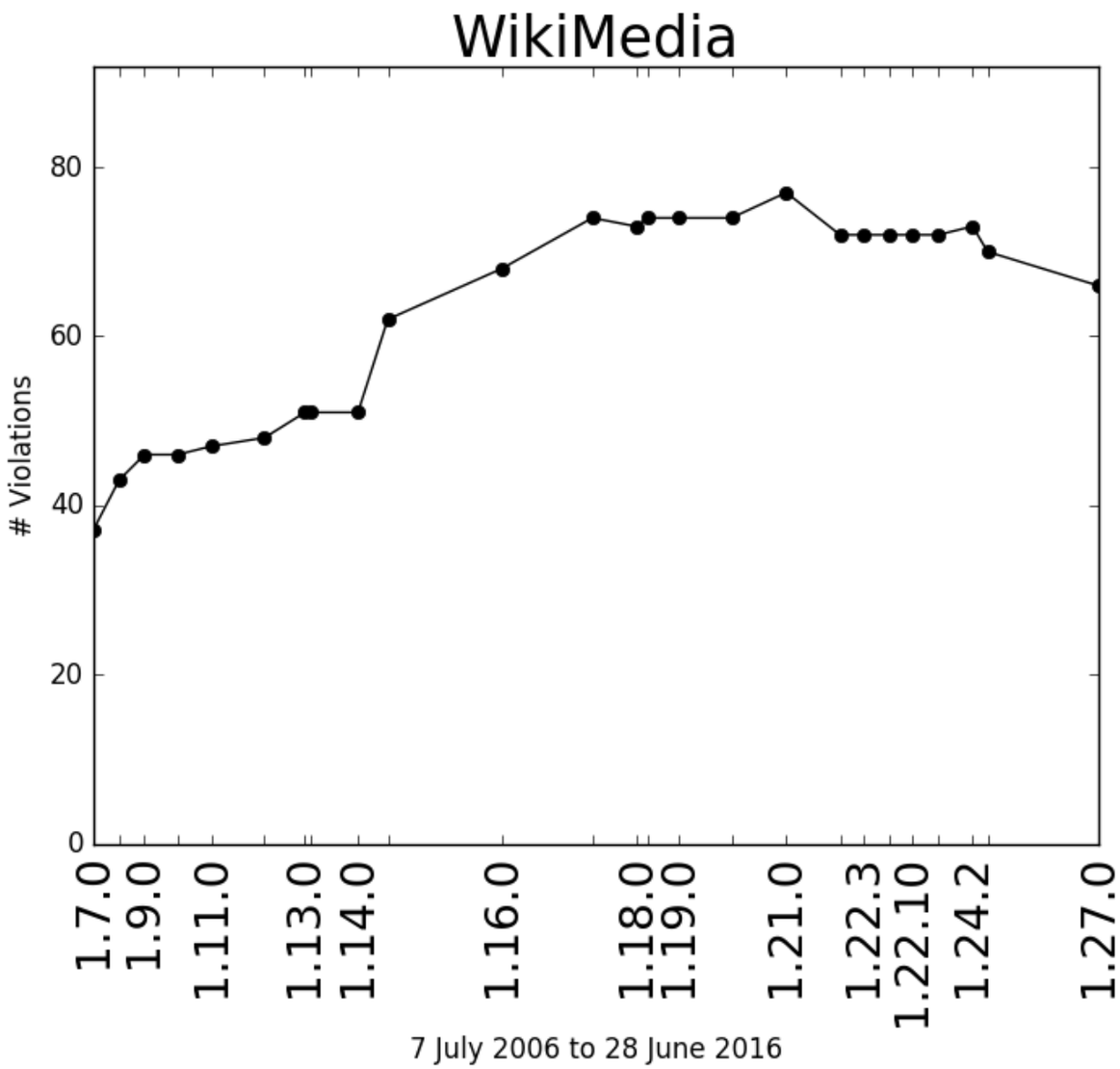
- AppSI: 12 versions analysed

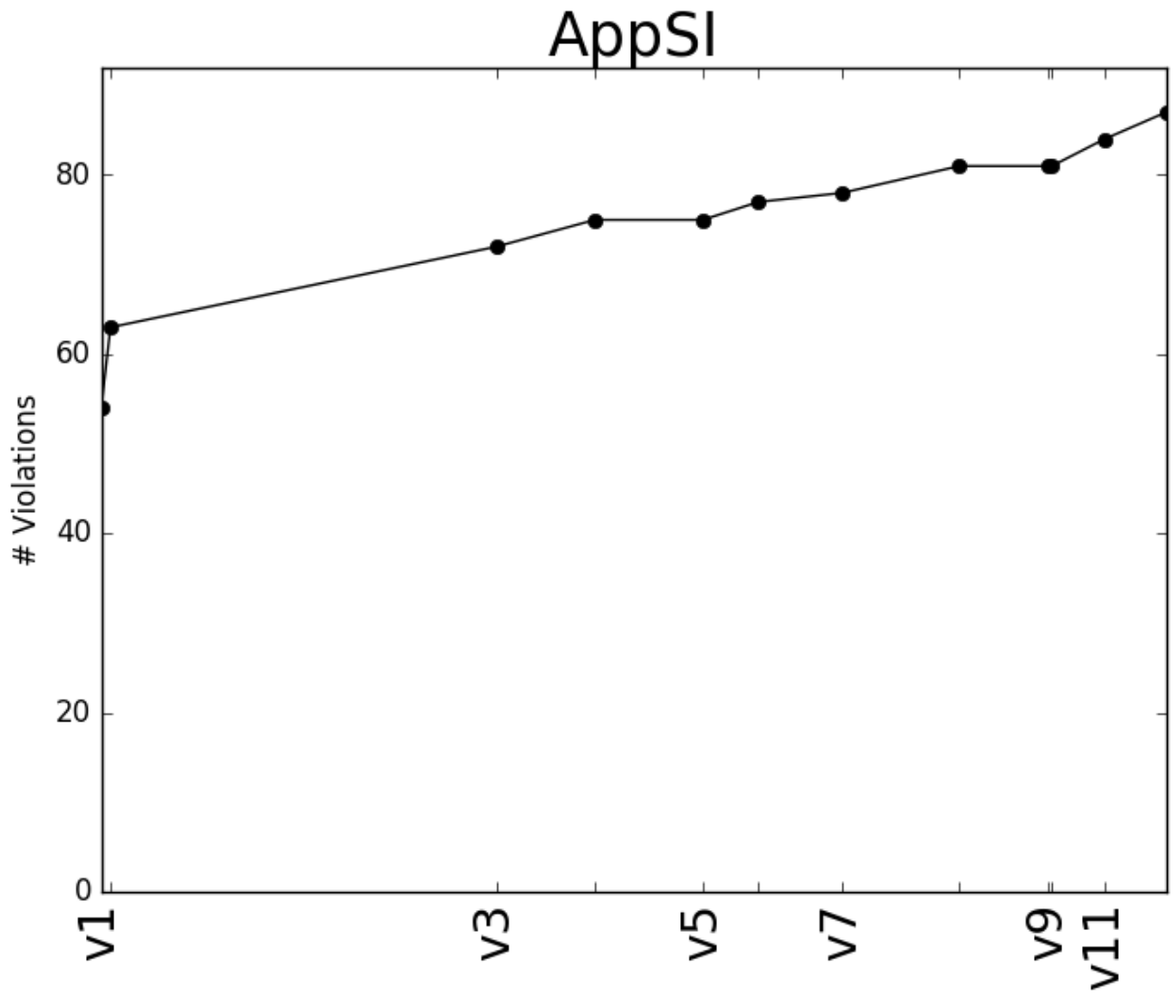
Proprietary database



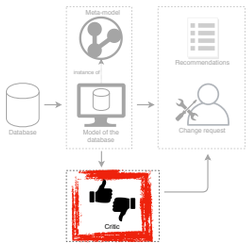


Violation count per version





Rule violations can be found in open source as well as in proprietary DB schemas.



Time-to-fix of a rule violation

Corrected violations:

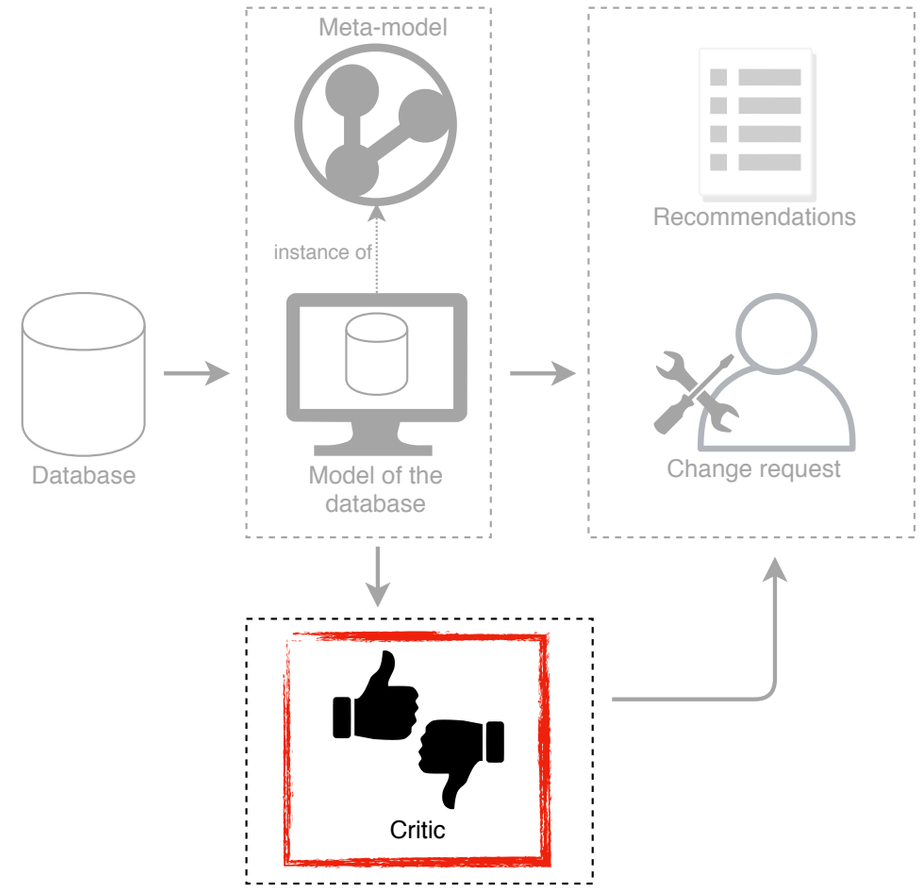
- WikiMedia: 21/87
- AppSI: 3/85

On both DBs some violations are fixed but not all of them.

	Min	First quantile	Median	Third quantile	Max
WikiMedia	95	1227	1833	2403	3644
AppSI	3	/	125	/	278

Time (in days) needed to correct violations

Summary

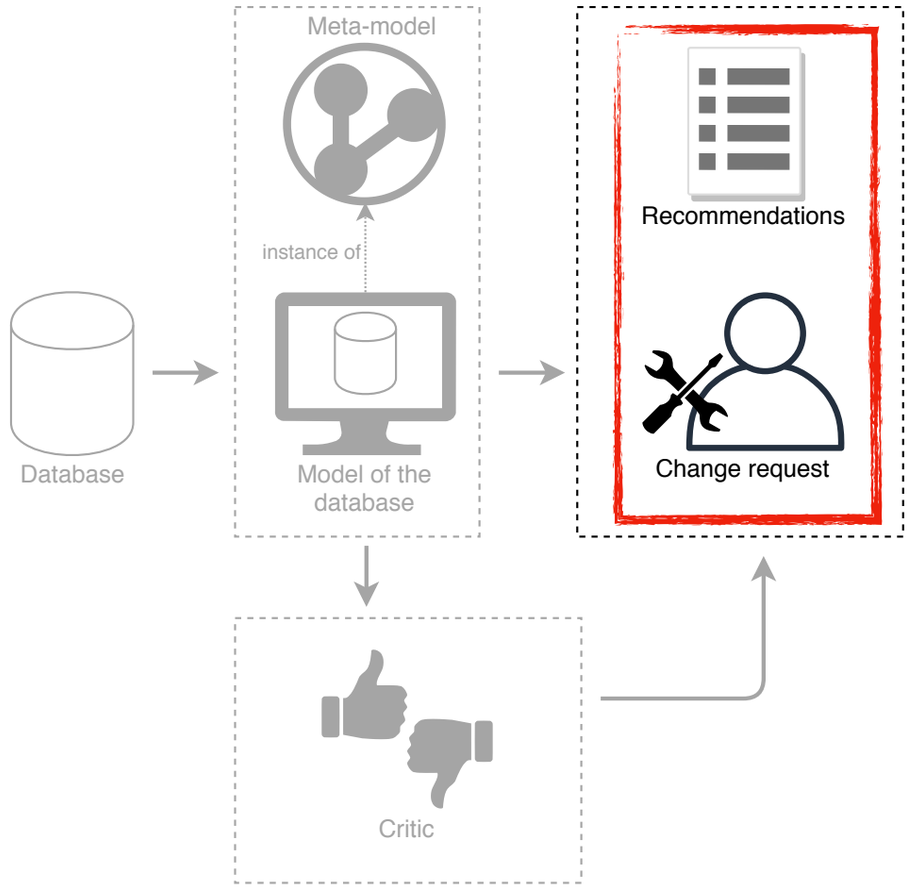


- **Rule violations can be found** in **open source** as well as in **proprietary** database schemas.
- The number of **violations increase with time.**
- With time, on both databases, **some violations are fixed but not all of them**.

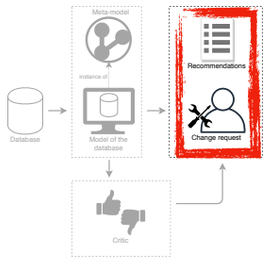
Roadmap

- Context & State of the Art
- Motivation
- A Behavior-Aware Meta-Model for Relational Databases
- Identifying Quality Issues in Relational Databases
- ▸ **Recommendations for Evolving Relational Databases**
- Conclusion

Recommendations for Evolving Relational Databases



Recommendations for Evolving Relational Databases. CAISE’20



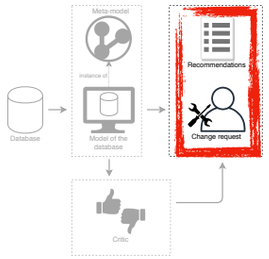
Example: remove beer.id

person

id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

beer

id	designation
1	Lupulus
2	Duvel
3	Troll



Example: remove beer.id

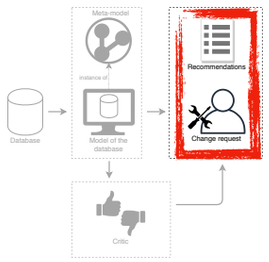
person

id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

beer

id	designation
1	Lupulus
2	Duvel
3	Troll

Forbidden by the RDBMS!



Example: remove column beer.designation referenced in view

person

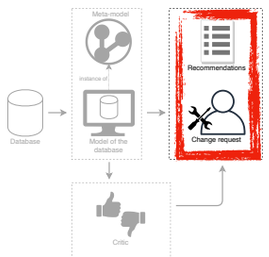
id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

beer

id	designation
1	Lupulus
2	Duvel
3	Troll

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```



Example: remove column beer.designation referenced in view

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```

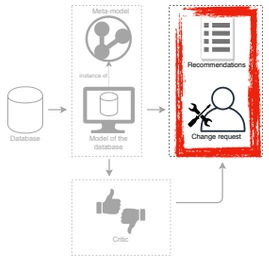
Refuse change

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```

⋮

Continues in cascade



Example: remove column beer.designation referenced in view

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id
```

Not convenient + behaviour

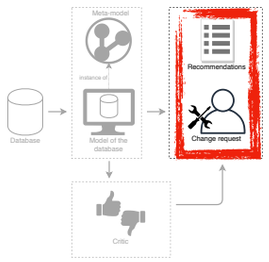
Refuse change

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```

⋮

Continues in cascade



Example: remove person.name

person

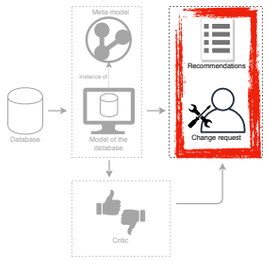
id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

beer

id	designation
1	Lupulus
2	Duvel
3	Troll

STOR. PROC. id_of_troll_lovers

```
RETURN SELECT person.id
FROM person, beer
WHERE person.favorite_beer = beer.id
    AND beer.designation = ‘Troll’;
```



Example: remove person.name

person

id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

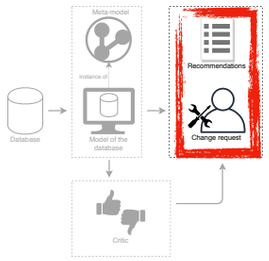
beer

id	designation
1	Lupulus
2	Duvel
3	Troll

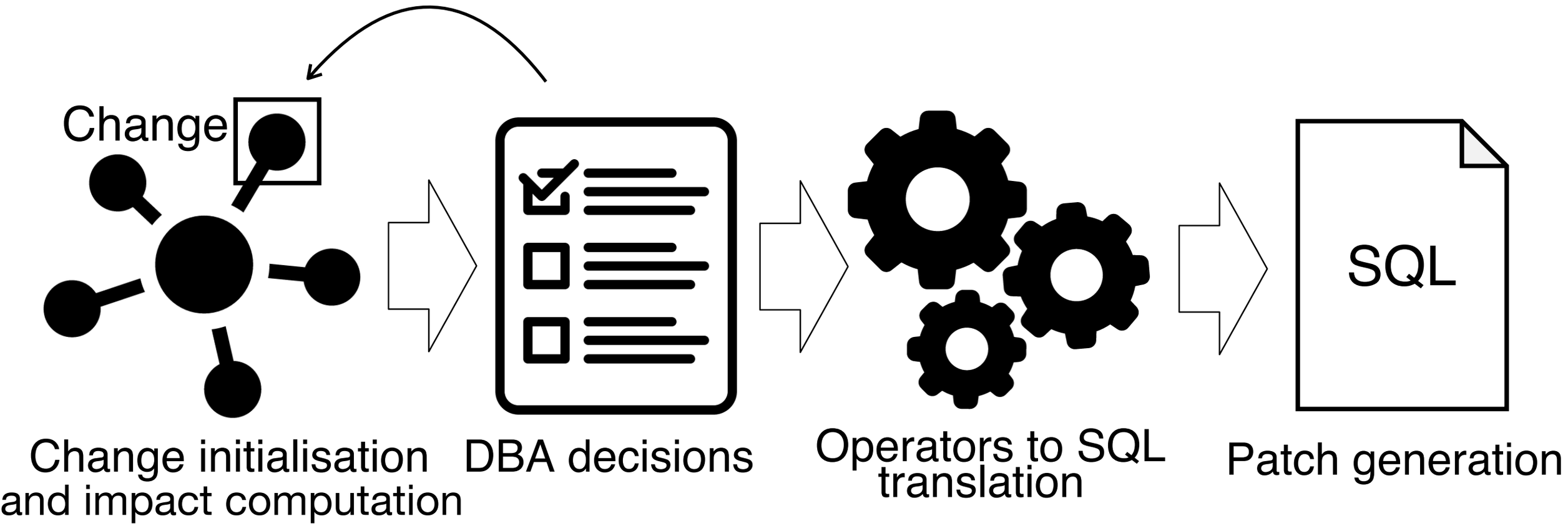
STOR. PROC. id_of_troll_lovers

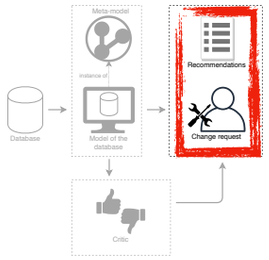
```
RETURN SELECT person.id
FROM person, beer
WHERE person.favorite_beer = beer.id
   AND beer.designation = ‘Troll’;
```

Unseen by the RDBMS!



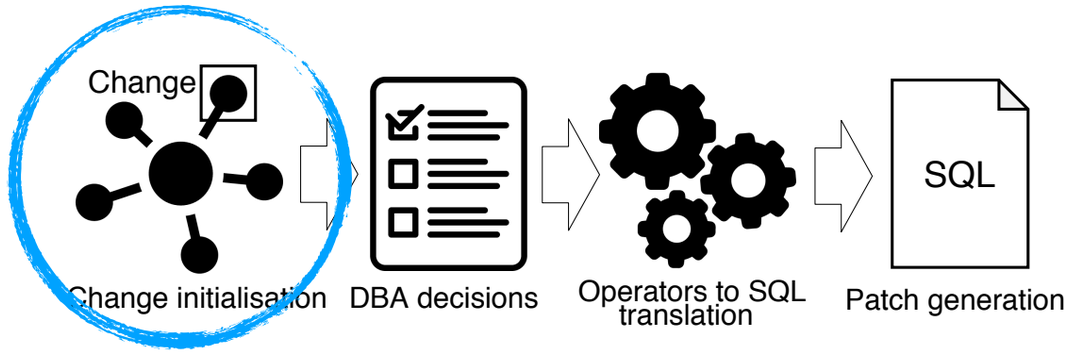
Approach overview

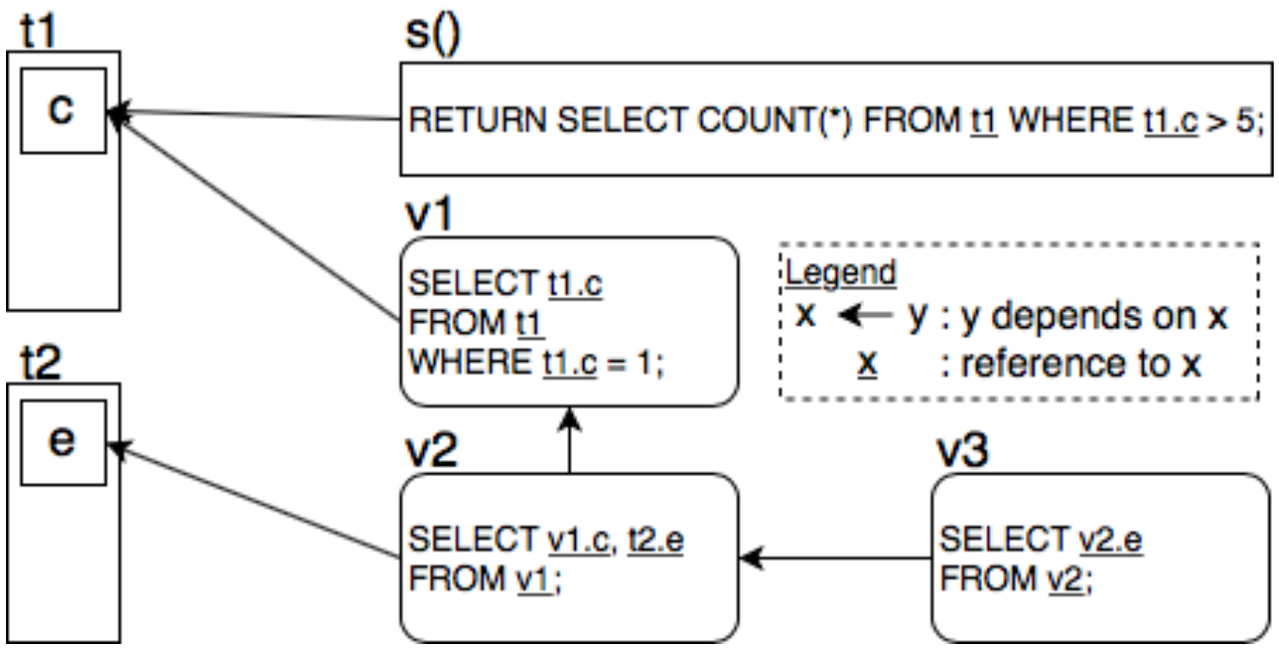


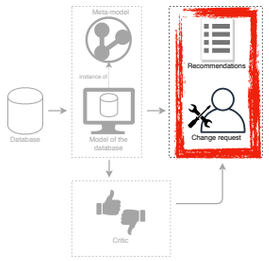


Model

You are here

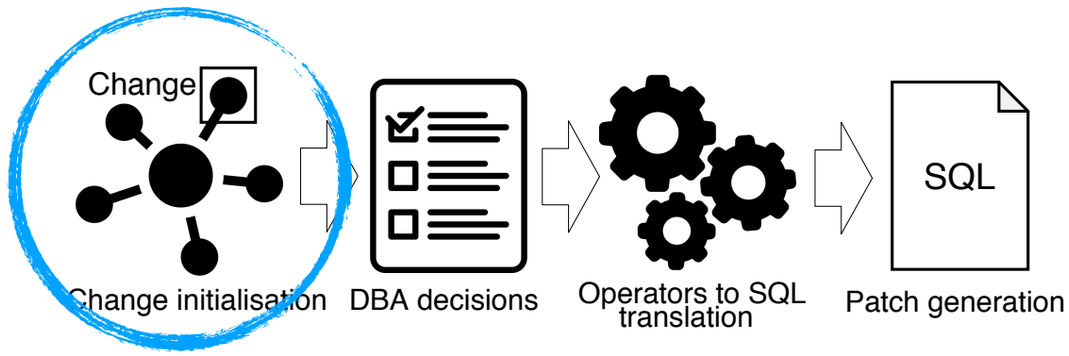


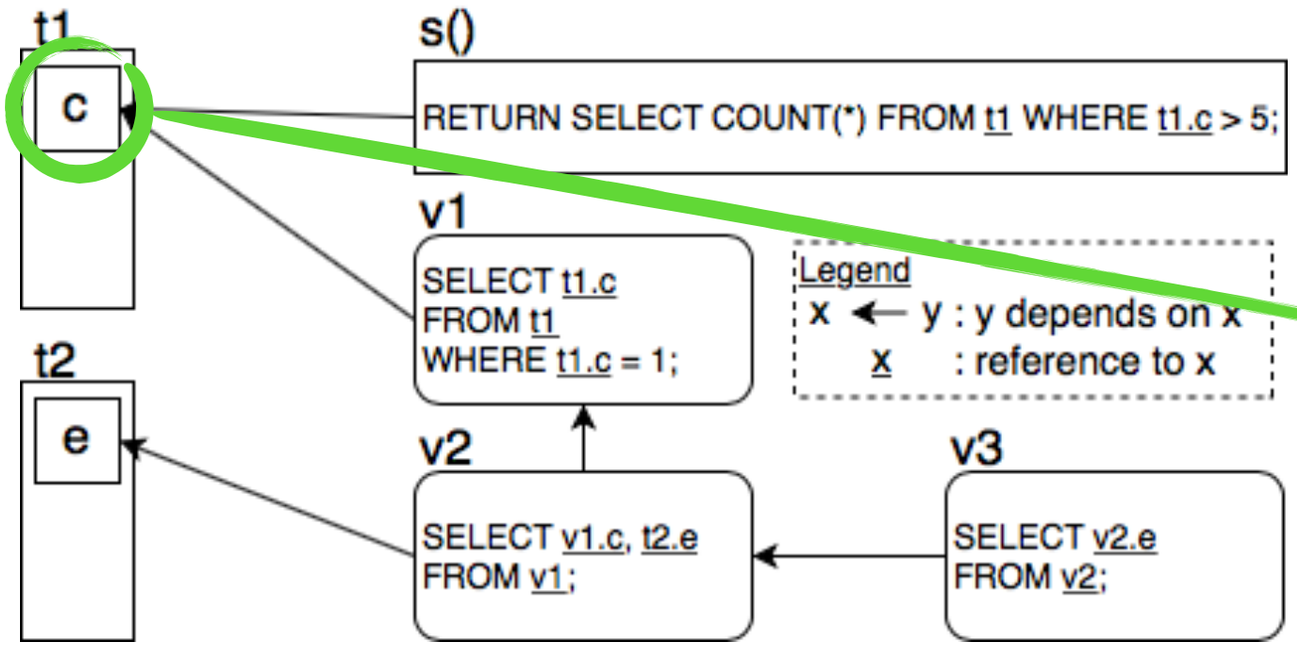




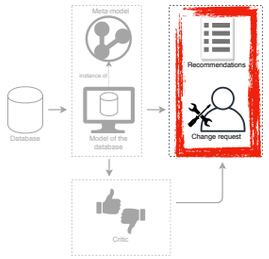
Model

You are here

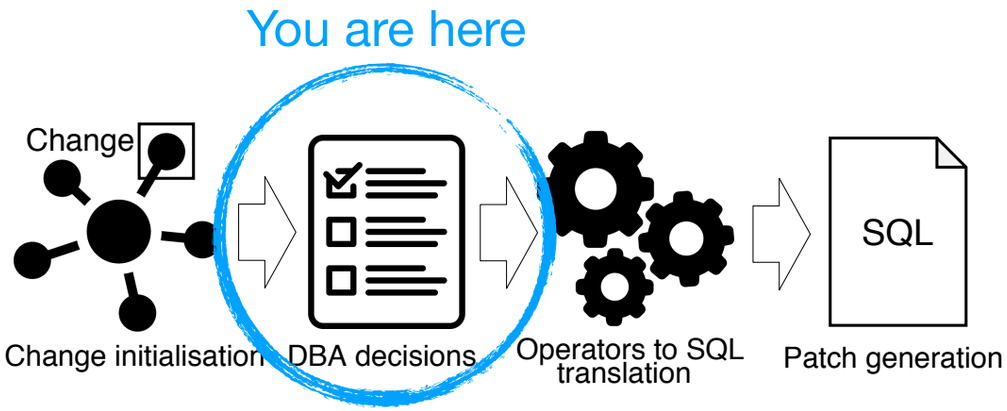


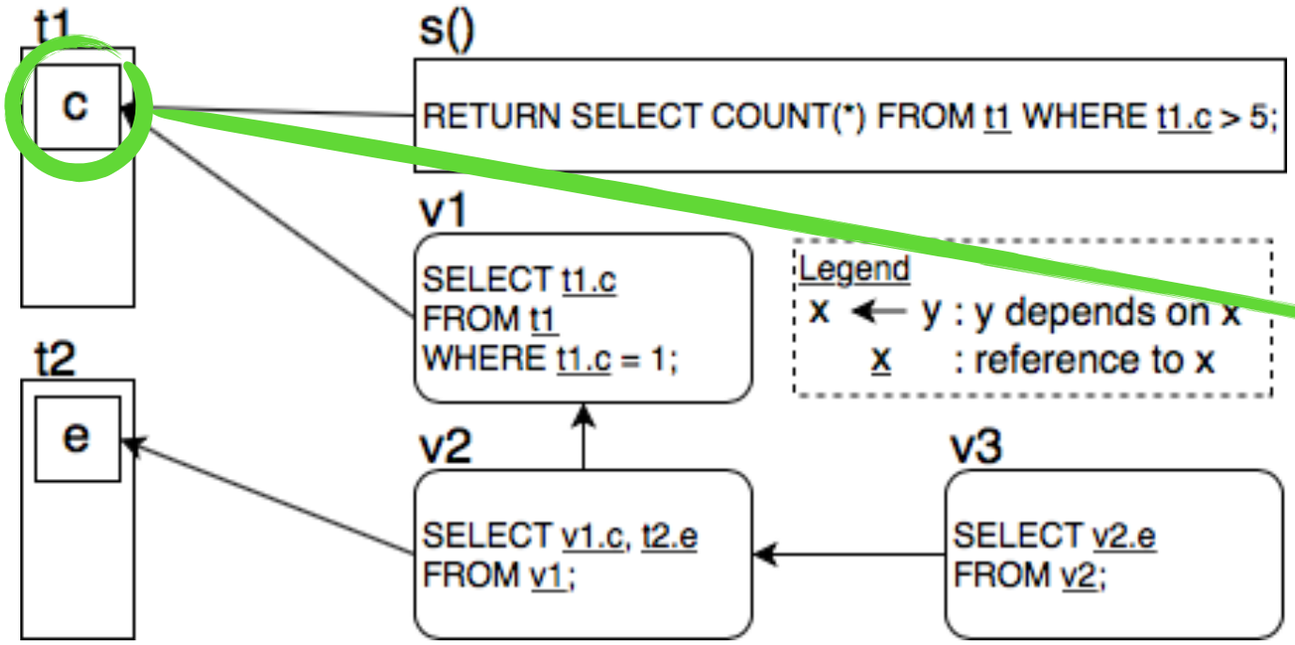


Rename t1.c column as t1.d

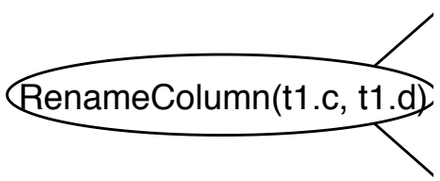


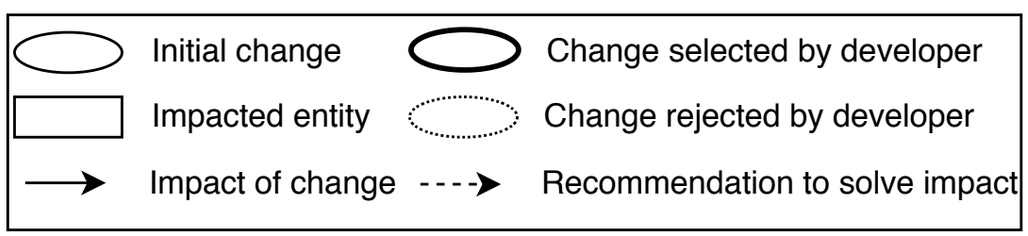
Impact computation & Recommendations selection

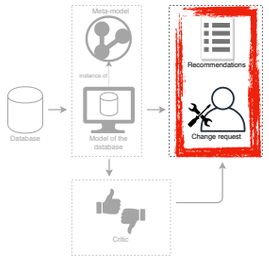




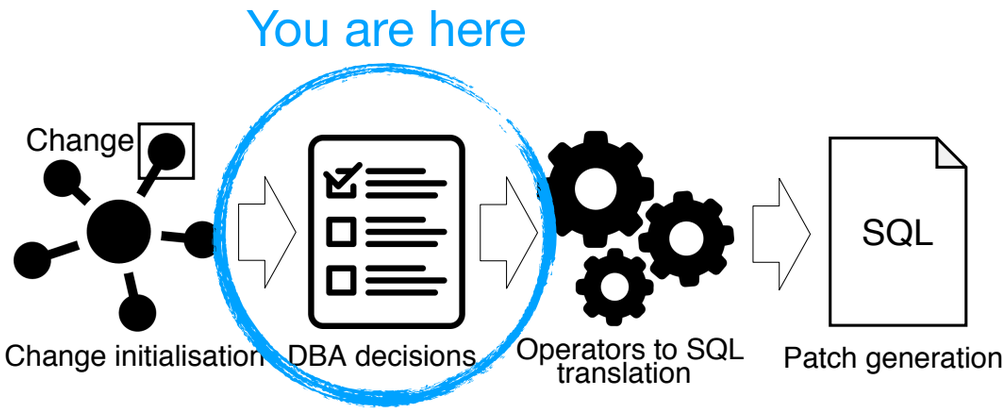
Rename t1.c column as t1.d

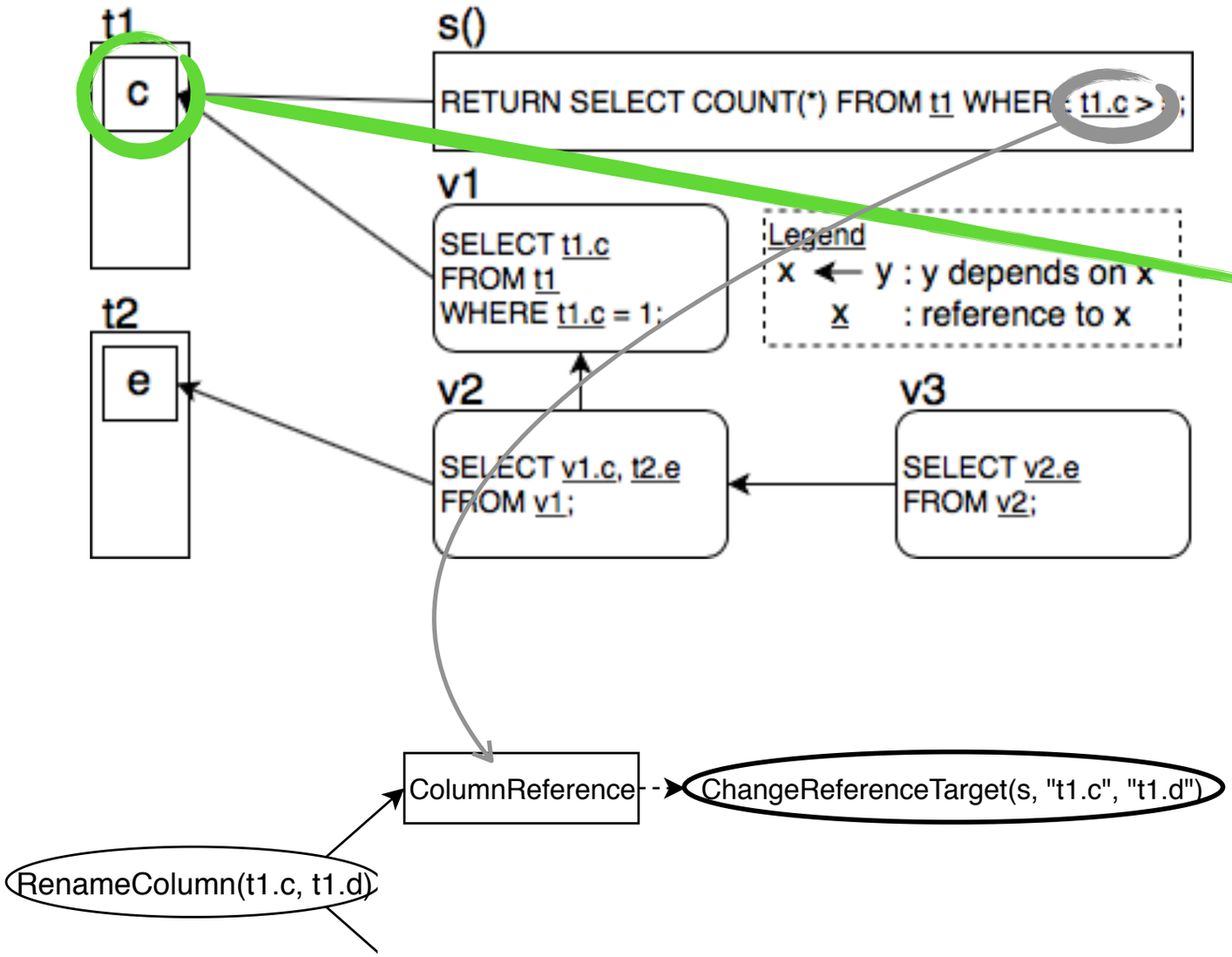




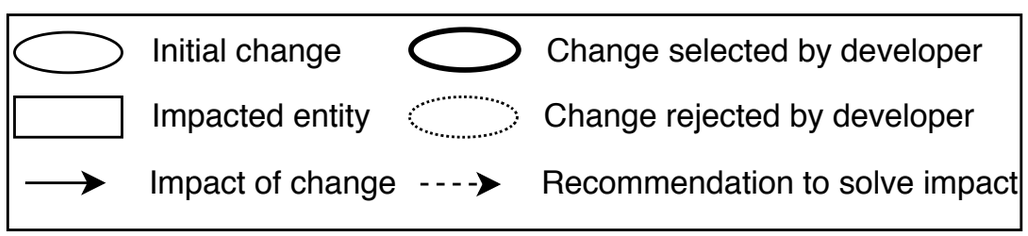


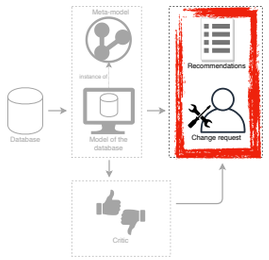
Impact computation & Recommendations selection



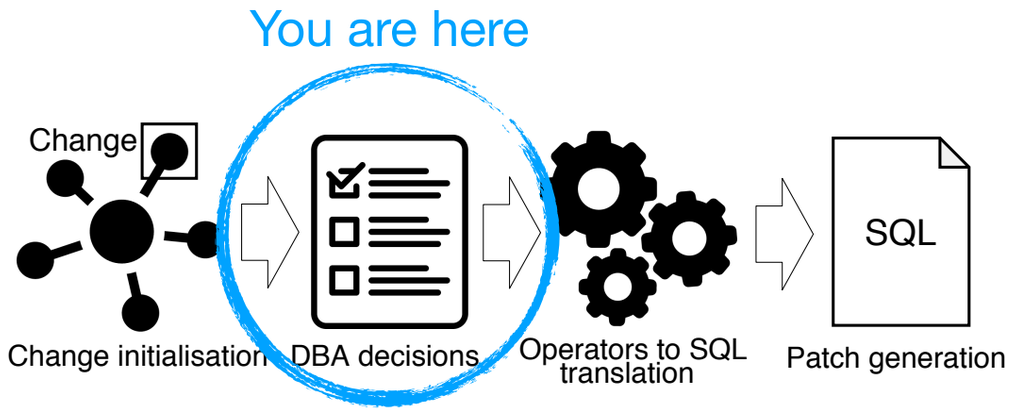


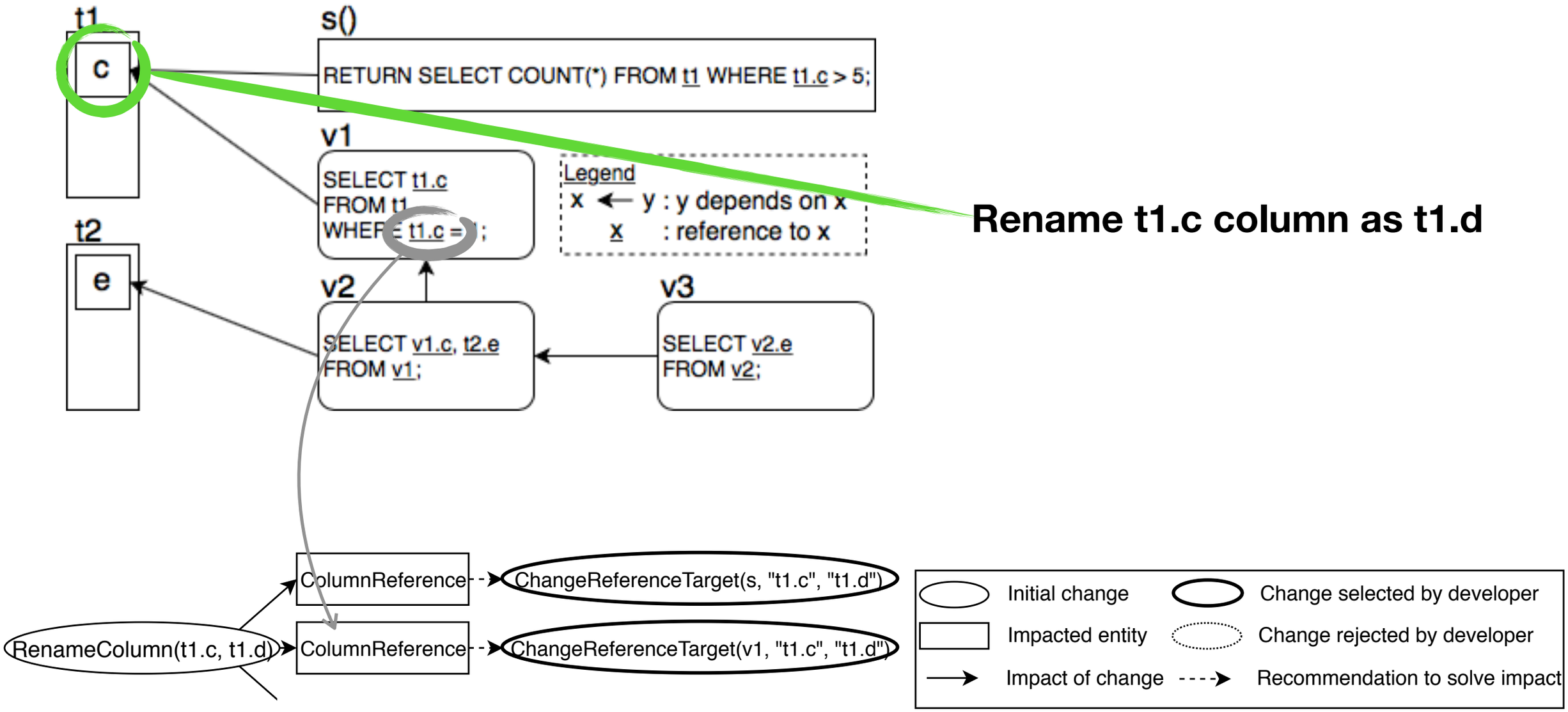
Rename t1.c column as t1.d

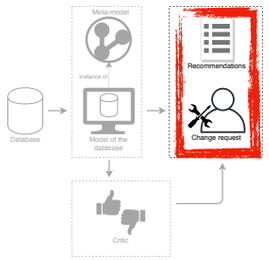




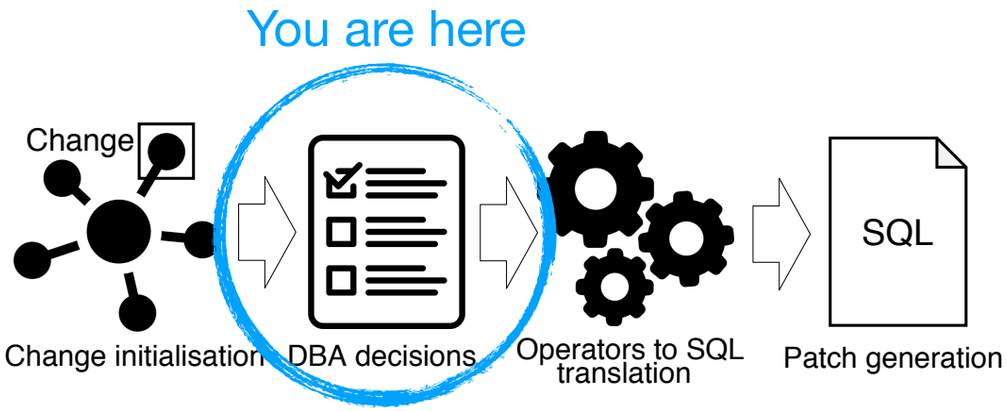
Impact computation & Recommendations selection

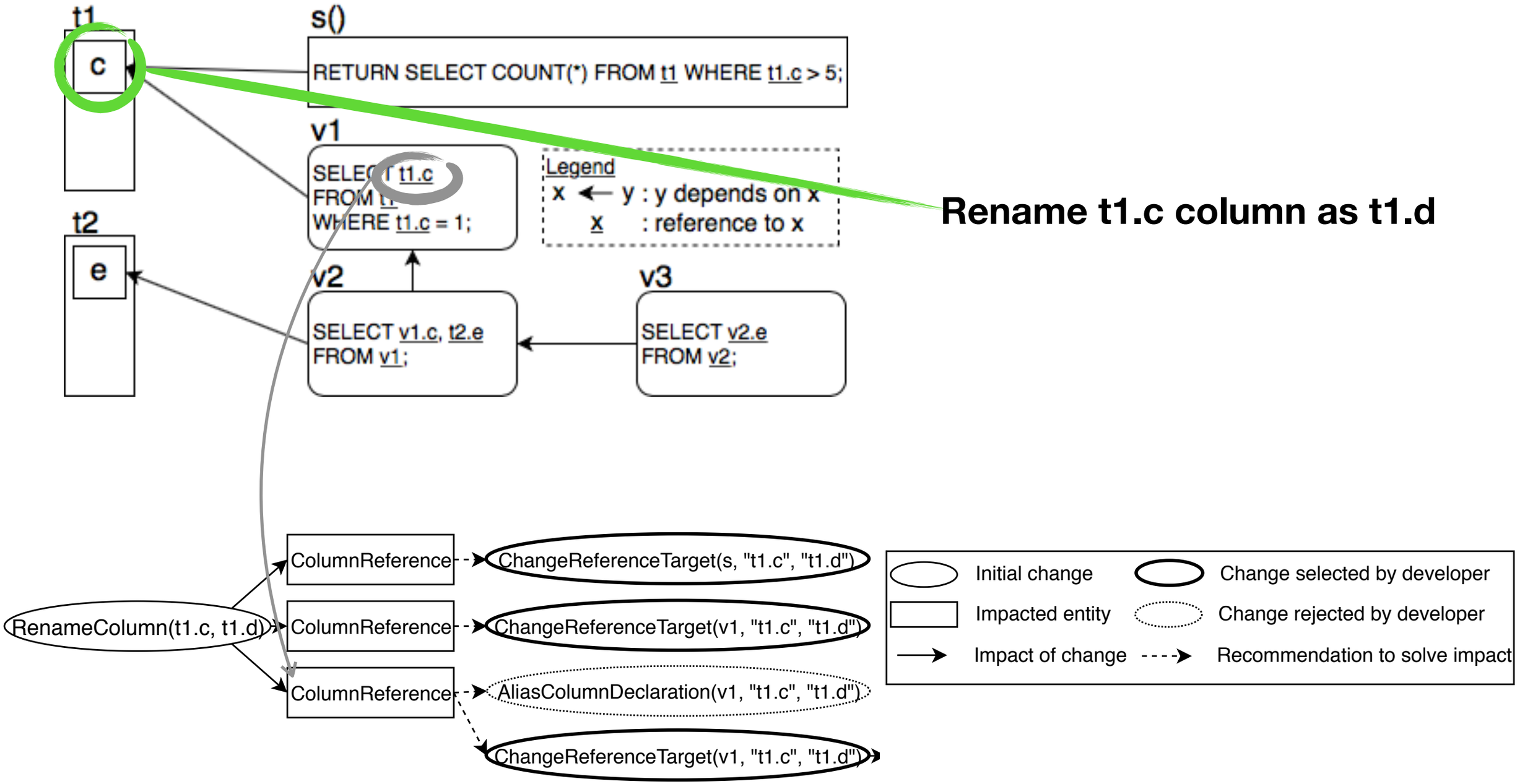


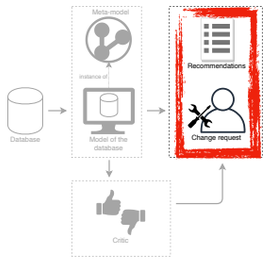




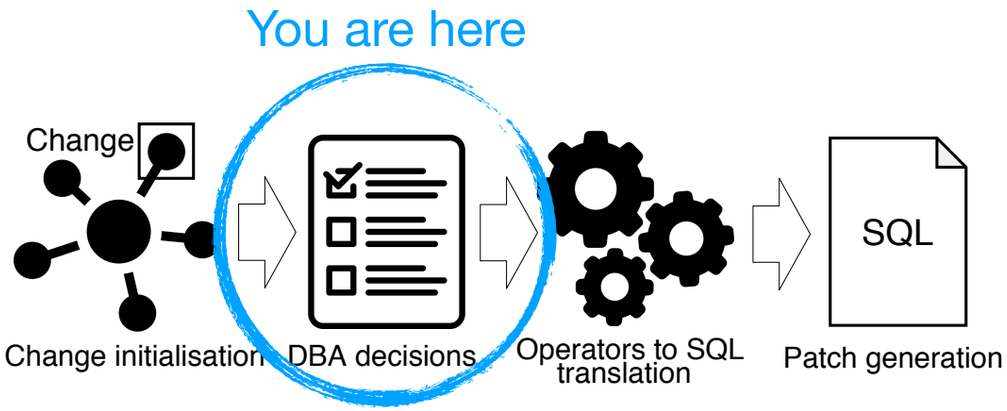
Impact computation & Recommendations selection

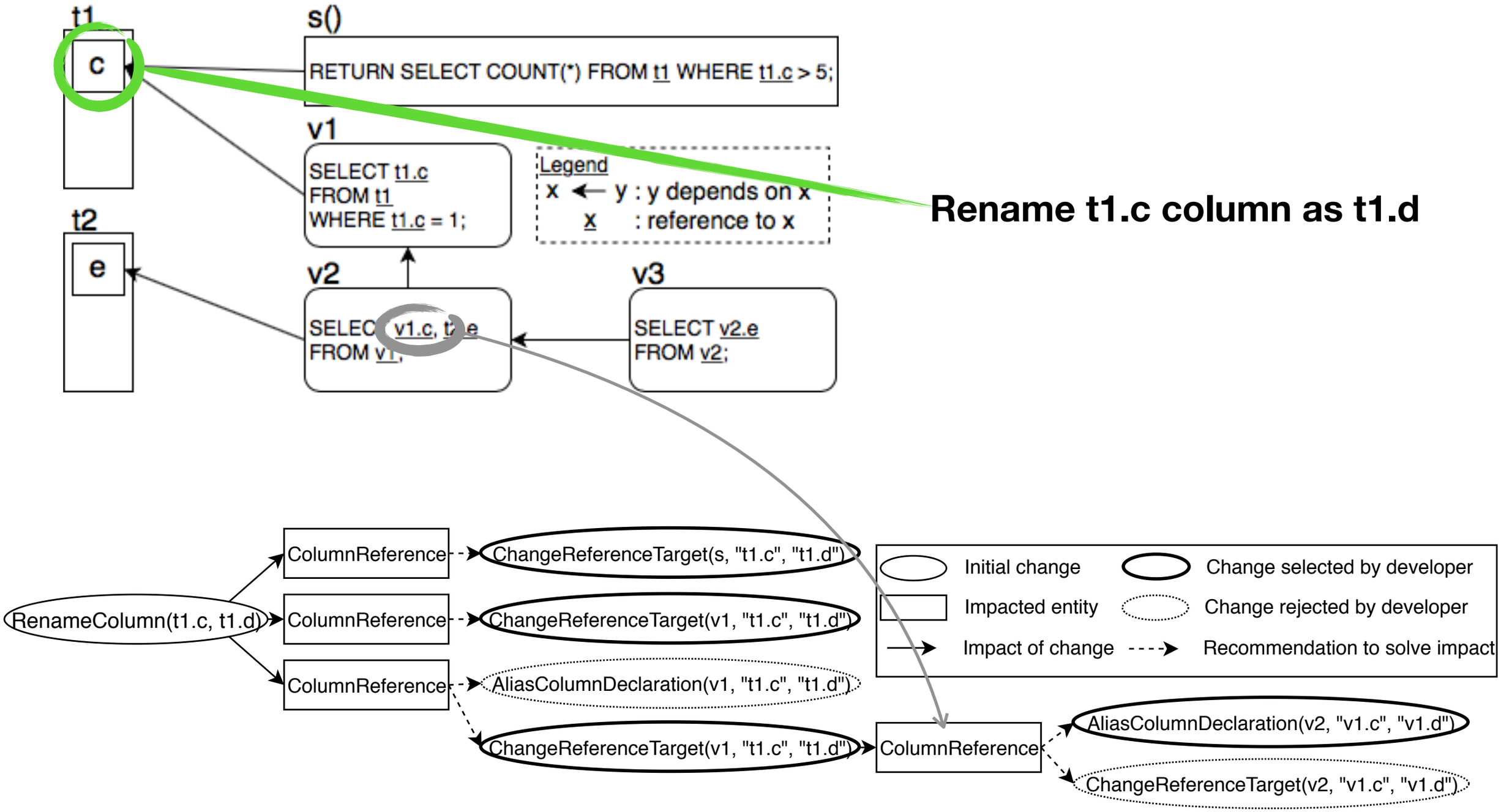


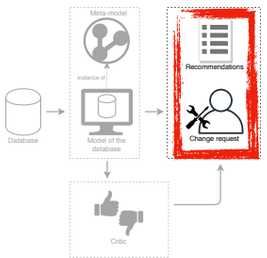




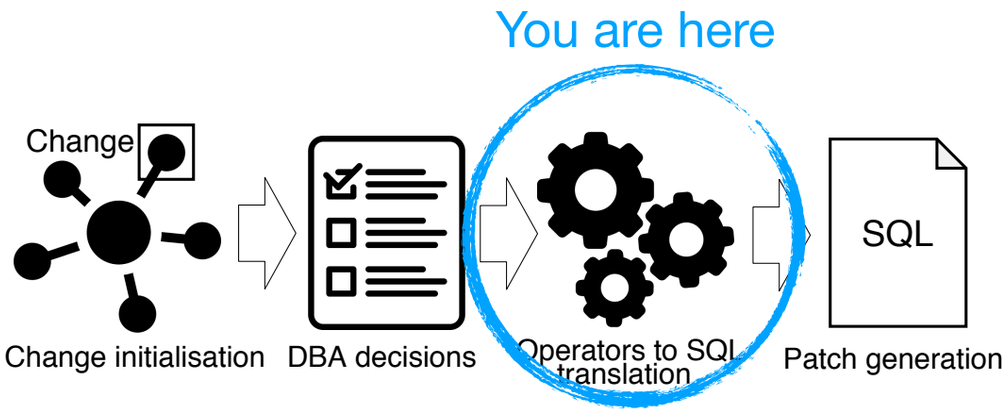
Impact computation & Recommendations selection

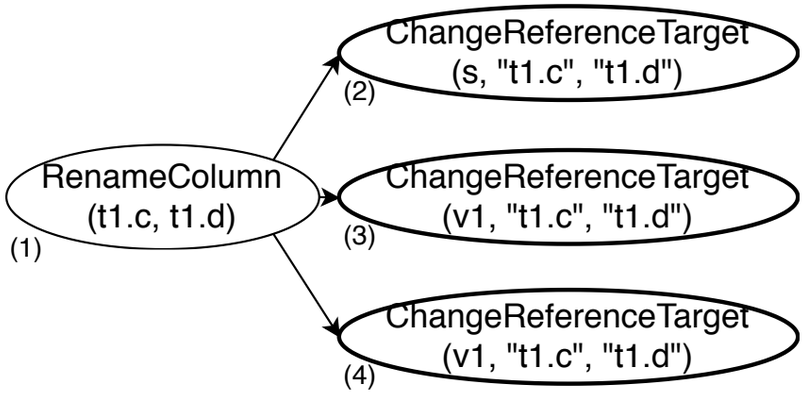


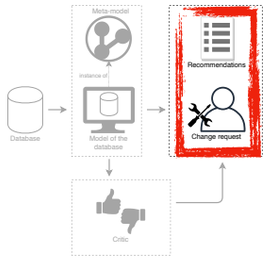




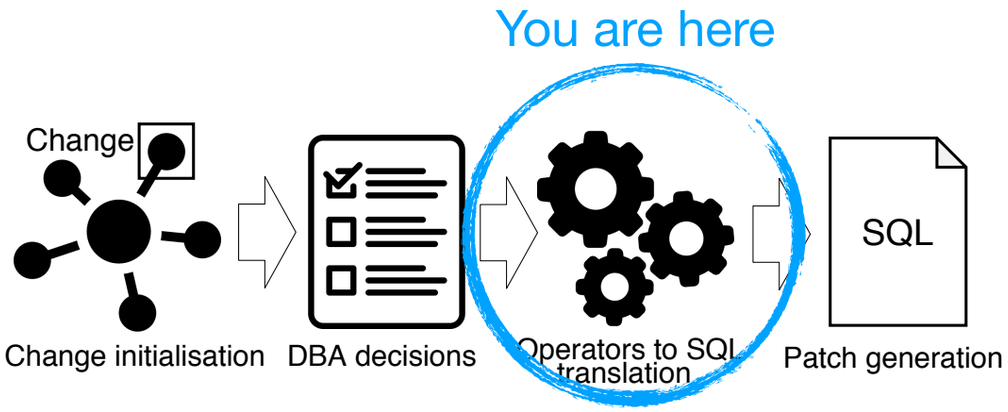
Architect choices compilation

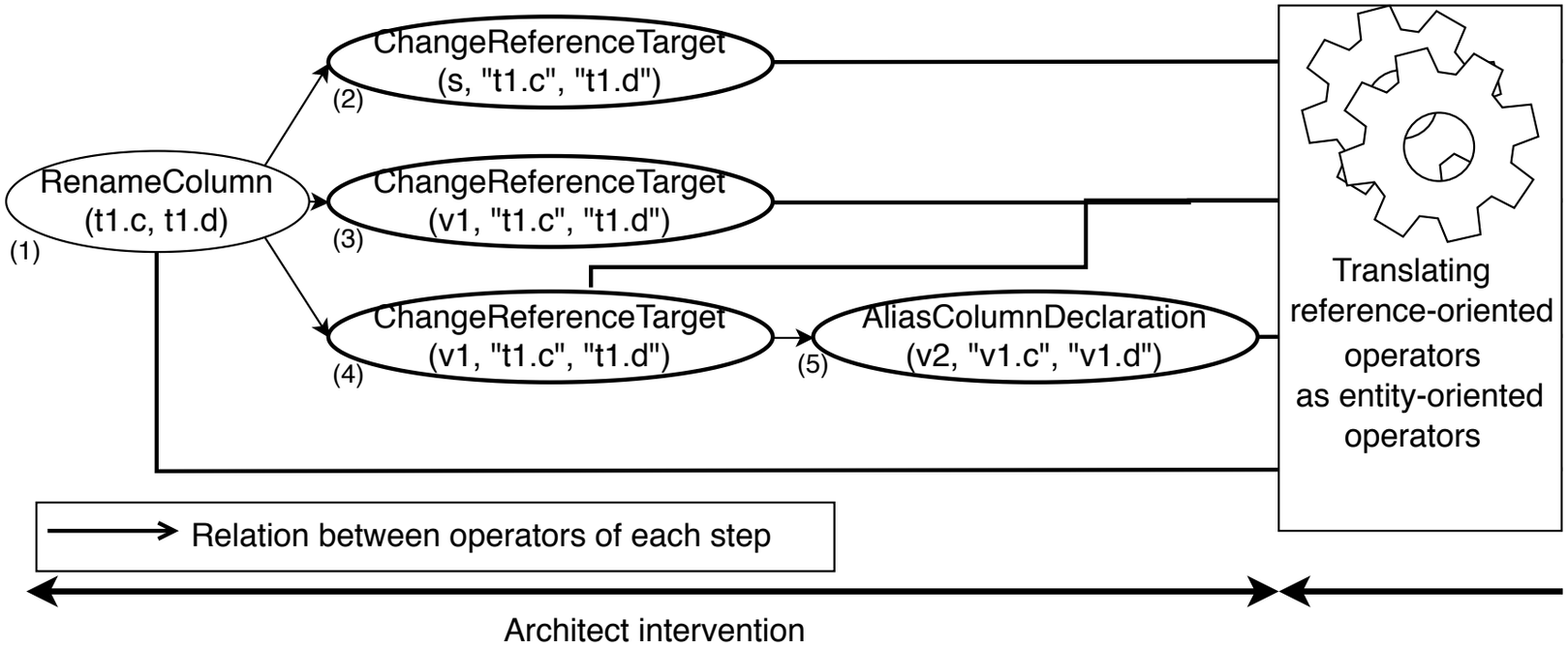


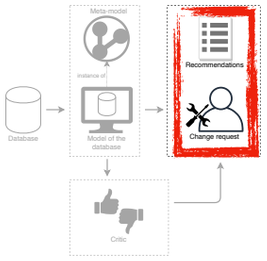




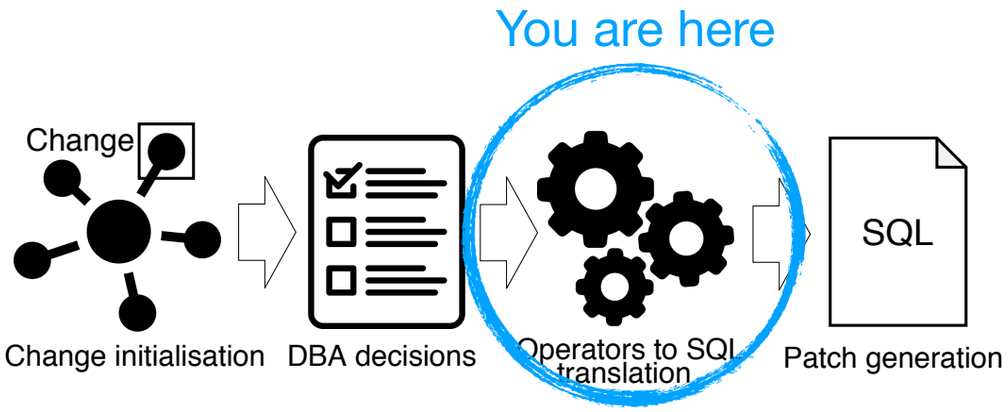
Architect choices compilation

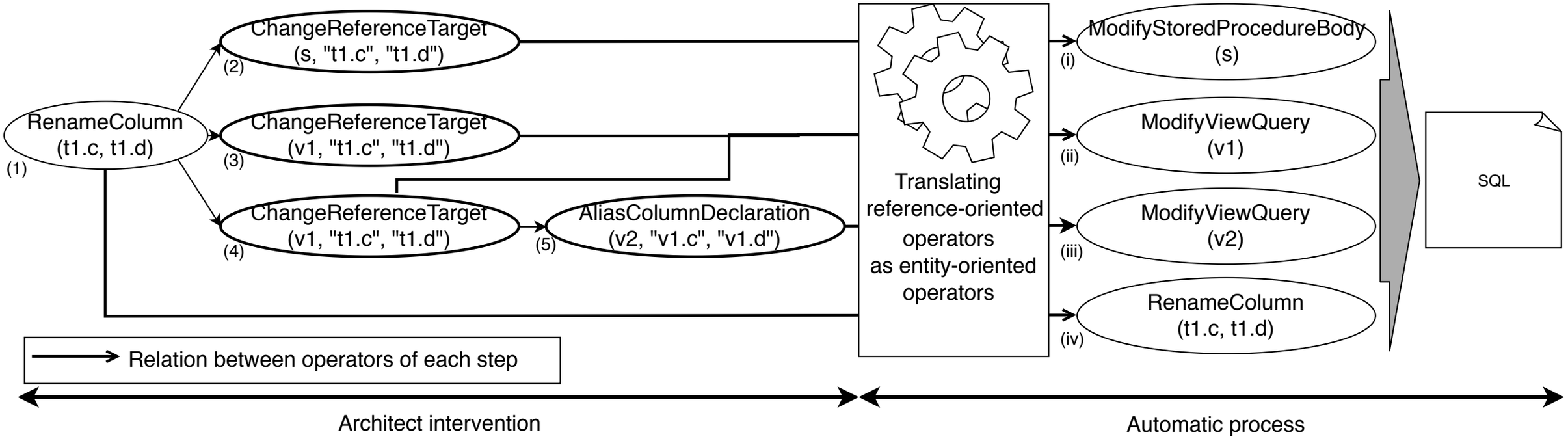


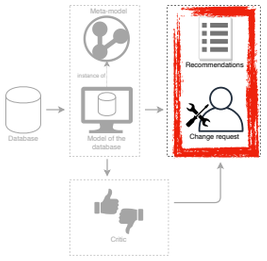




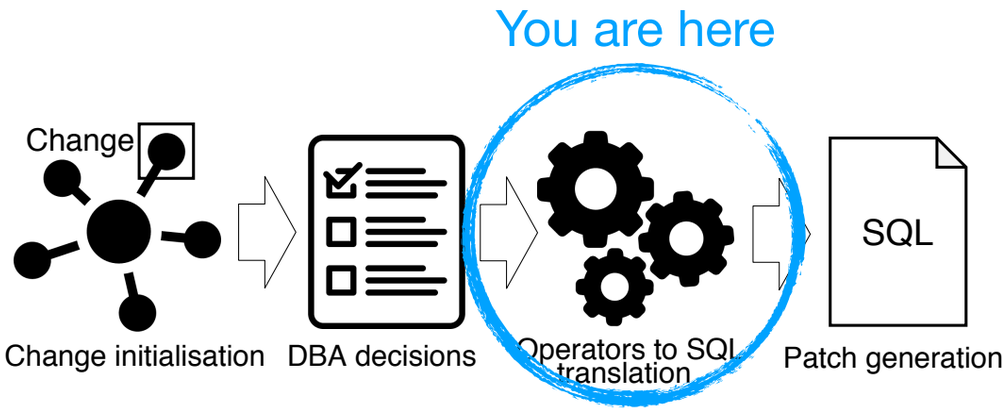
Architect choices compilation

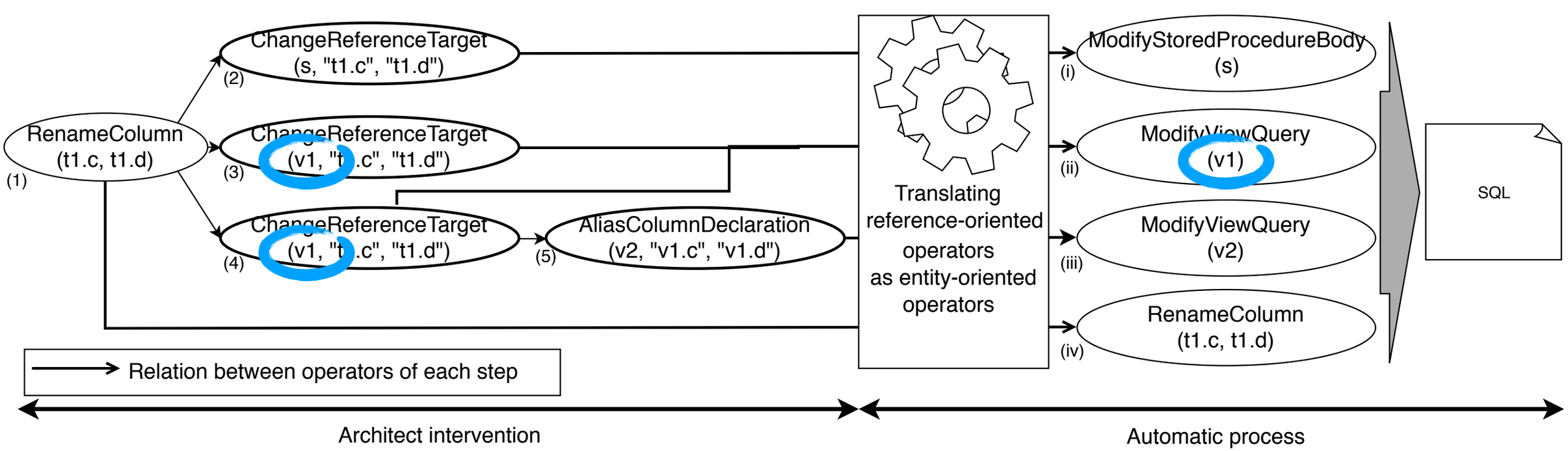


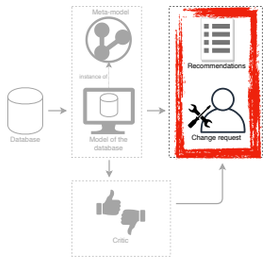




Architect choices compilation





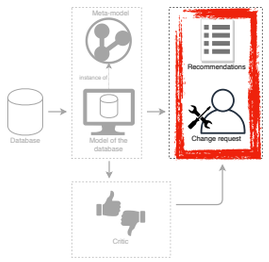


Experiment

- **Post-mortem analysis** of a **SQL patch** previously applied on AppSI

- We observed **trial-and-error process** from DBA to find dependencies between entities in motivation part

- **1h** to achieve a script of **200 LOC** / **19 statements**

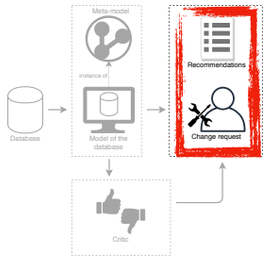


Experimental protocol

1. **Extract initial** semantic **operators** from roadmap and SQL script comments

2. Take decisions using **DBA's policy** when multiple possibilities (change reference target or alias reference)

3. **Execute** generated **SQL** script on a database in the same state as information system's **database before the migration**



Results

- **15 decisions** taken concerning the choice between changing reference target and aliasing reference.

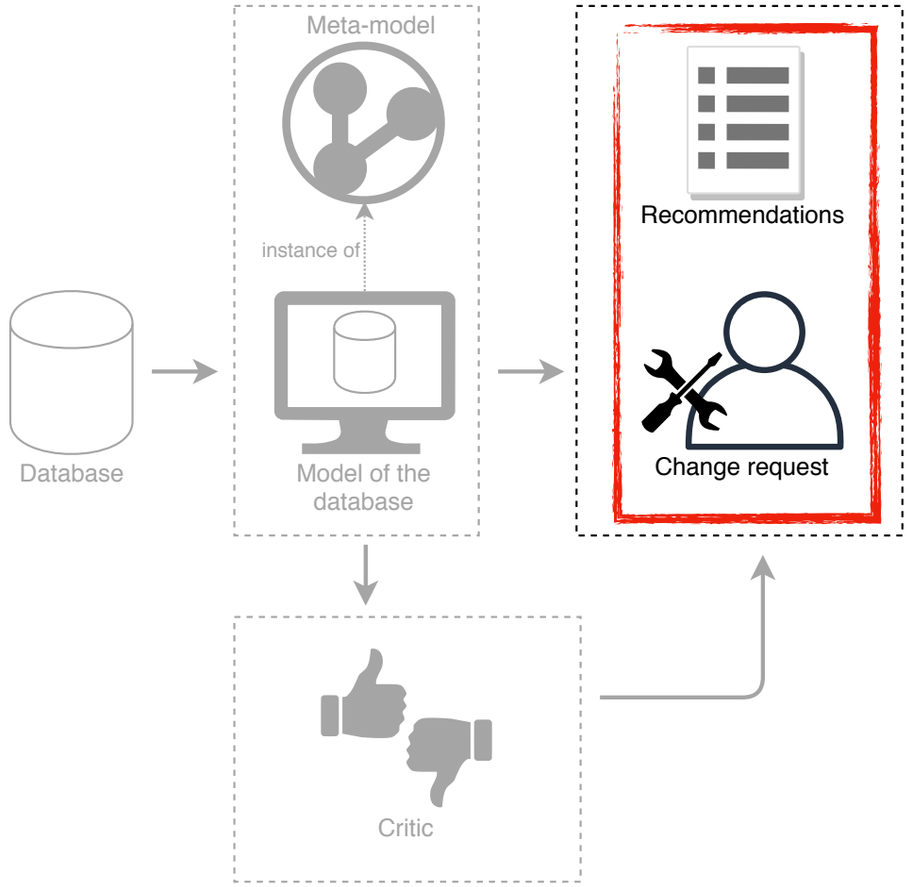
- **270 LOC** script with **27 SQL statements**.

- Generated script **executed without error**.

- **Single difference** between the dump of the original database and our clone: **a comment**.

- **Time to implement** the evolution using our tool: **15 min** (v.s. 60 min without tool).

Summary



- Semi-automatic **approach to address « Relational database are hard to evolve » problem**

- **Experiment** to assess the effectiveness of our approach

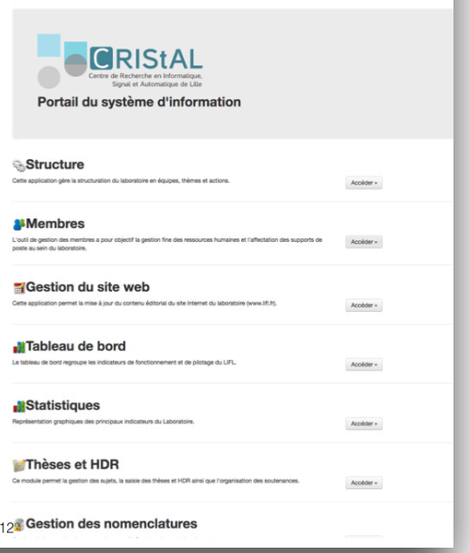
Roadmap

- Context & State of the Art
- Motivation
- A Behavior-Aware Meta-Model for Relational Databases
- Identifying Quality Issues in Relational Databases
- Recommendations for Evolving Relational Databases
- ▸ **Conclusion**

We identified problems occurring during relational database evolution…

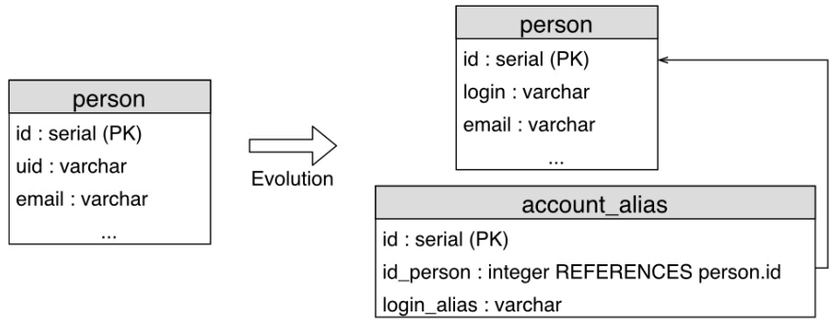
AppSI database overview

- Members, teams, funding support, etc. management in laboratories of our university;
- Used by software systems written in different programming languages;
- Used in multiple laboratories at the university.



12

An evolution of AppSI

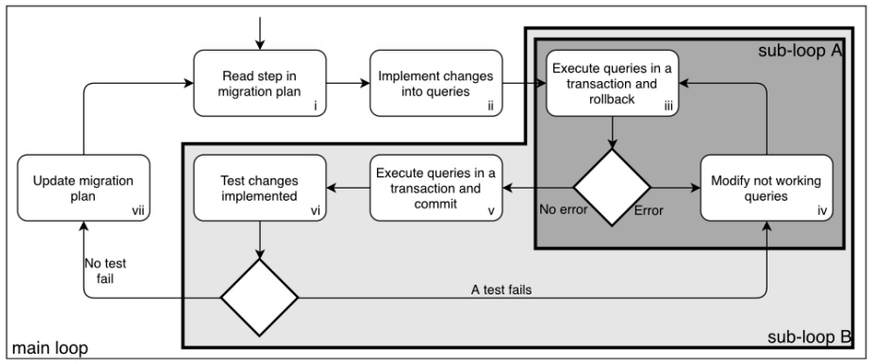


After the modification,

- Allow users to have multiple identifiers ➡ *account_alias* table is created
- Gather all the secondary identifiers of a *person* ➡ *login_alias* column

15

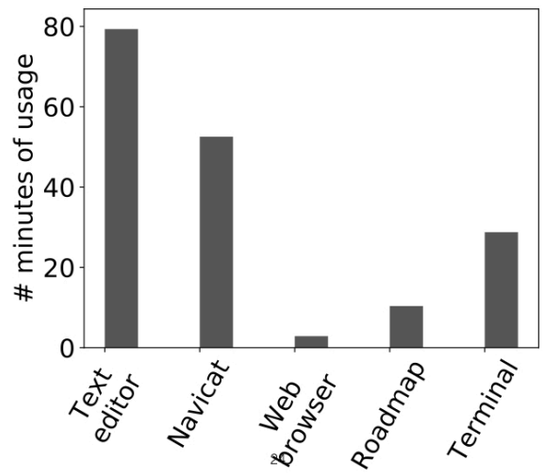
Qualitative Analysis



- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.
- **main loop**: complete implementation of a feature (might include multiple iterations on loop A and/or loop B).

21

Quantitative Analysis



We identified problems occurring during relational database evolution…

AppSI database overview

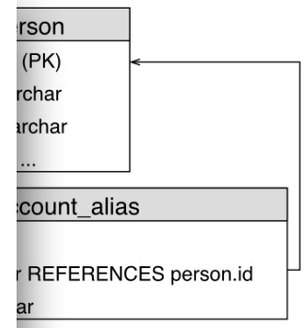
- Members, teams, funding
  support, etc. manageme
  laboratories of our univer
- Used by software system
  written in different
  programming languages;
- Used in multiple laborato
  the university.

Problems

- **Relational databases are hard to understand.**
  - RDBMS meta-data are complex to query.
  - Some meta-data are missing.
- **Relational databases are hard to evolve.**
  - Dependencies between a database and a program using it are usually implicit.
  - The database schema can not be in an inconsistent state at any moment.

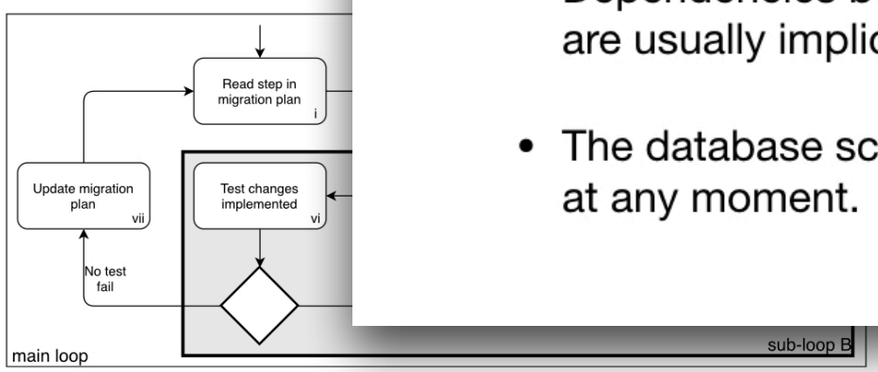
28

An evolution of AppSI



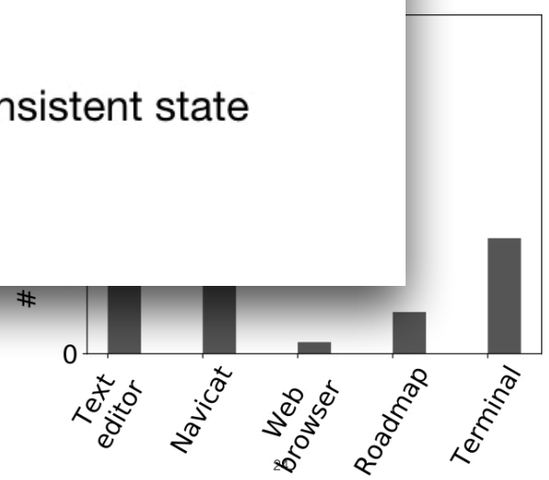
nt_alias table is created
➡ login_alias column

Qualit… Analysis

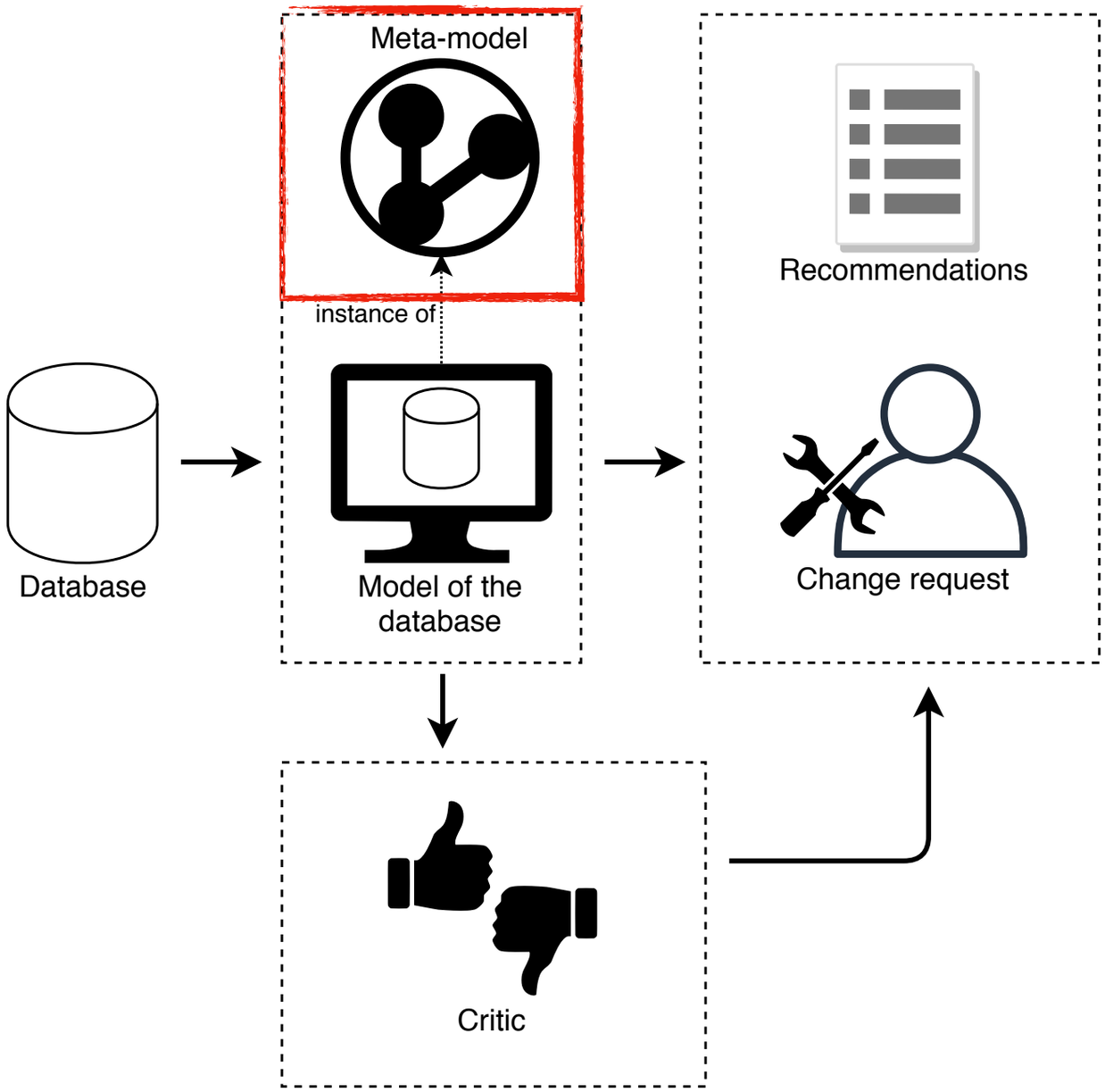


- **sub-loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **sub-loop B**: resolution of semantic errors.
- **main loop**: complete implementation of a feature (might include multiple iterations on loop A and/or loop B).

21



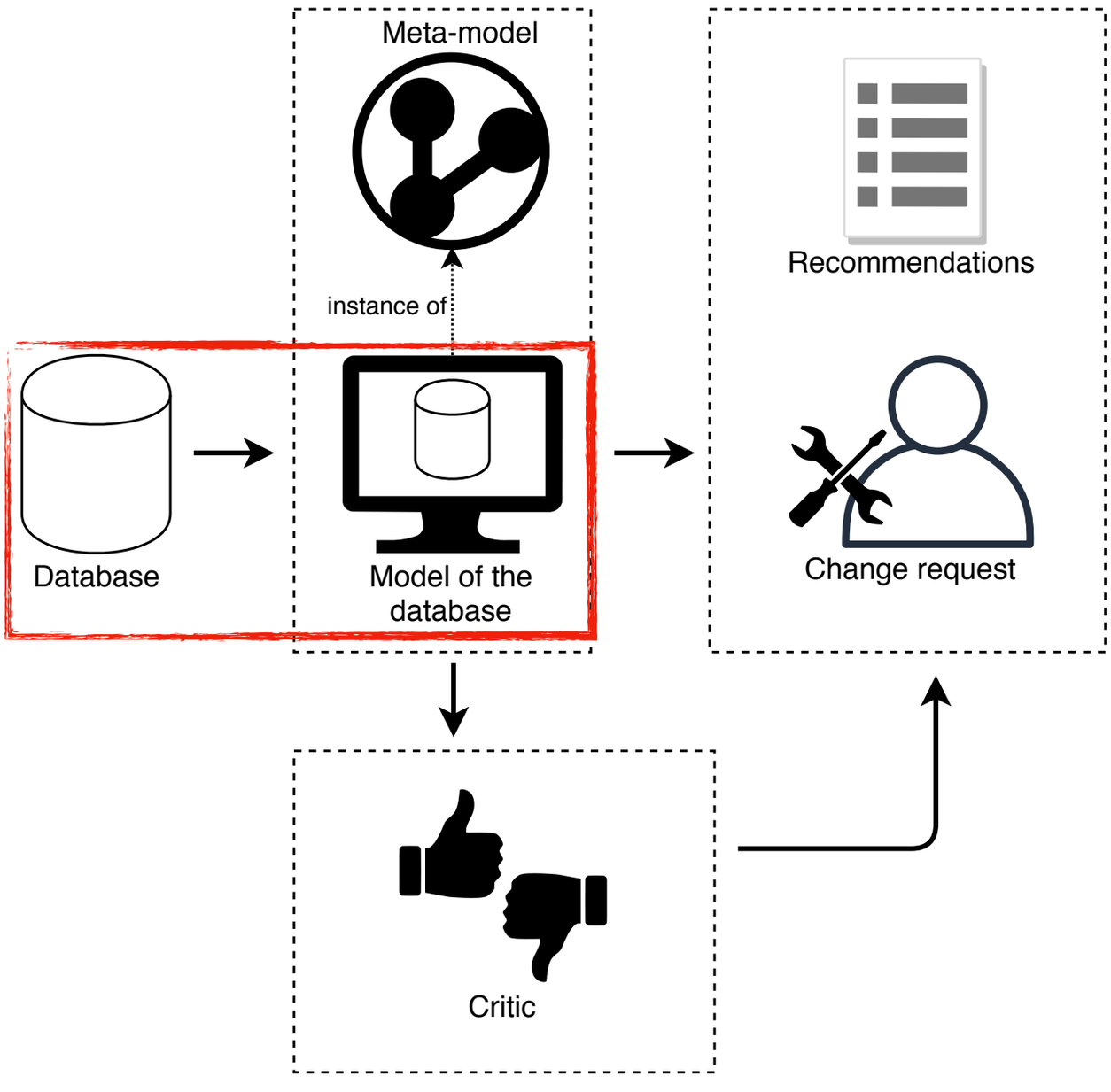
… and designed an approach to tackle it.



Including:

- **a meta-model**
- an approach to build it
- a tool to assess DB quality
- an approach to evolve DB
- empirical experiment(s) for each part

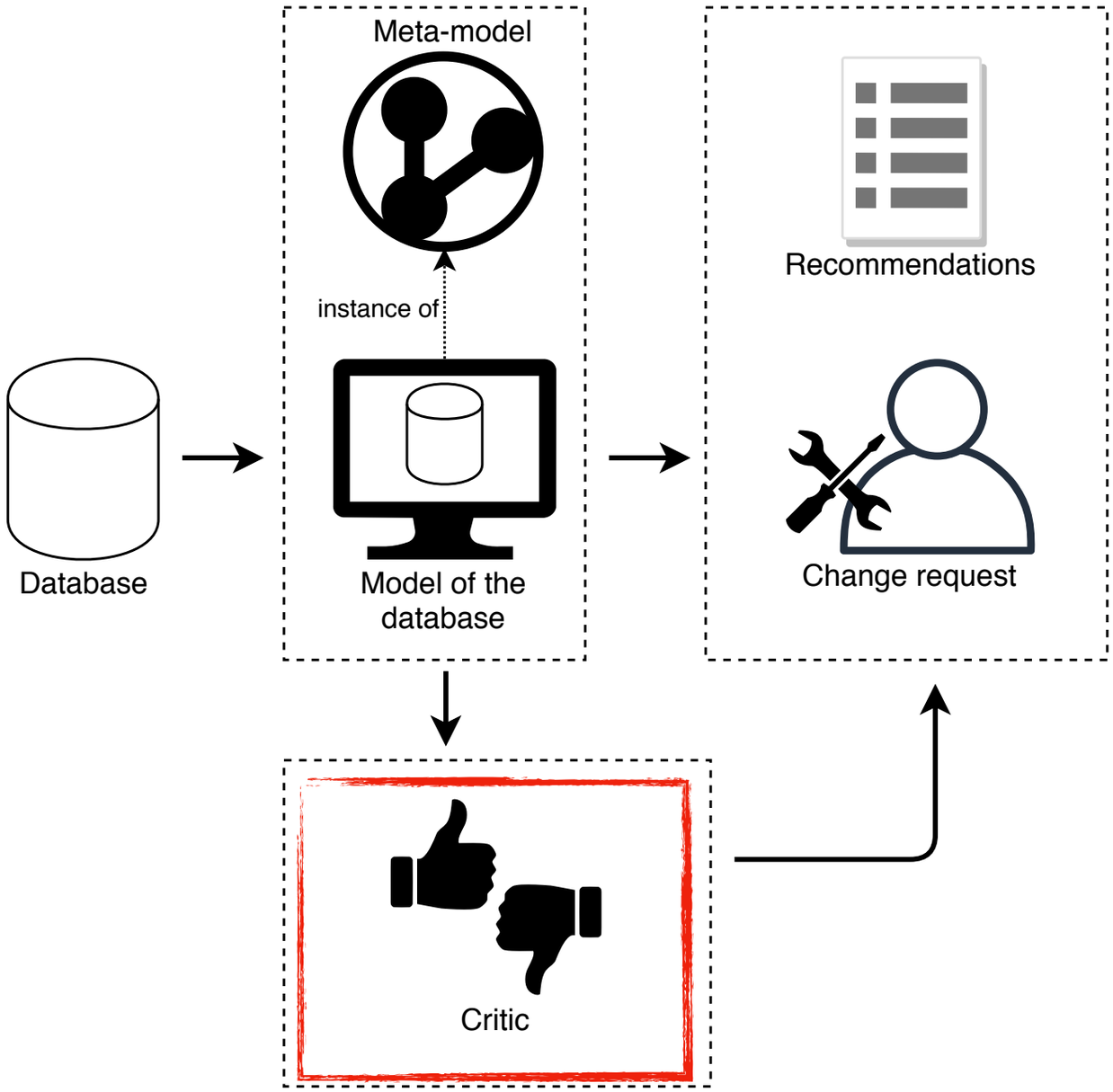
… and designed an approach to tackle it.



Including:

- a meta-model
- **an approach to build it**
- a tool to assess DB quality
- an approach to evolve DB
- empirical experiment(s) for each part

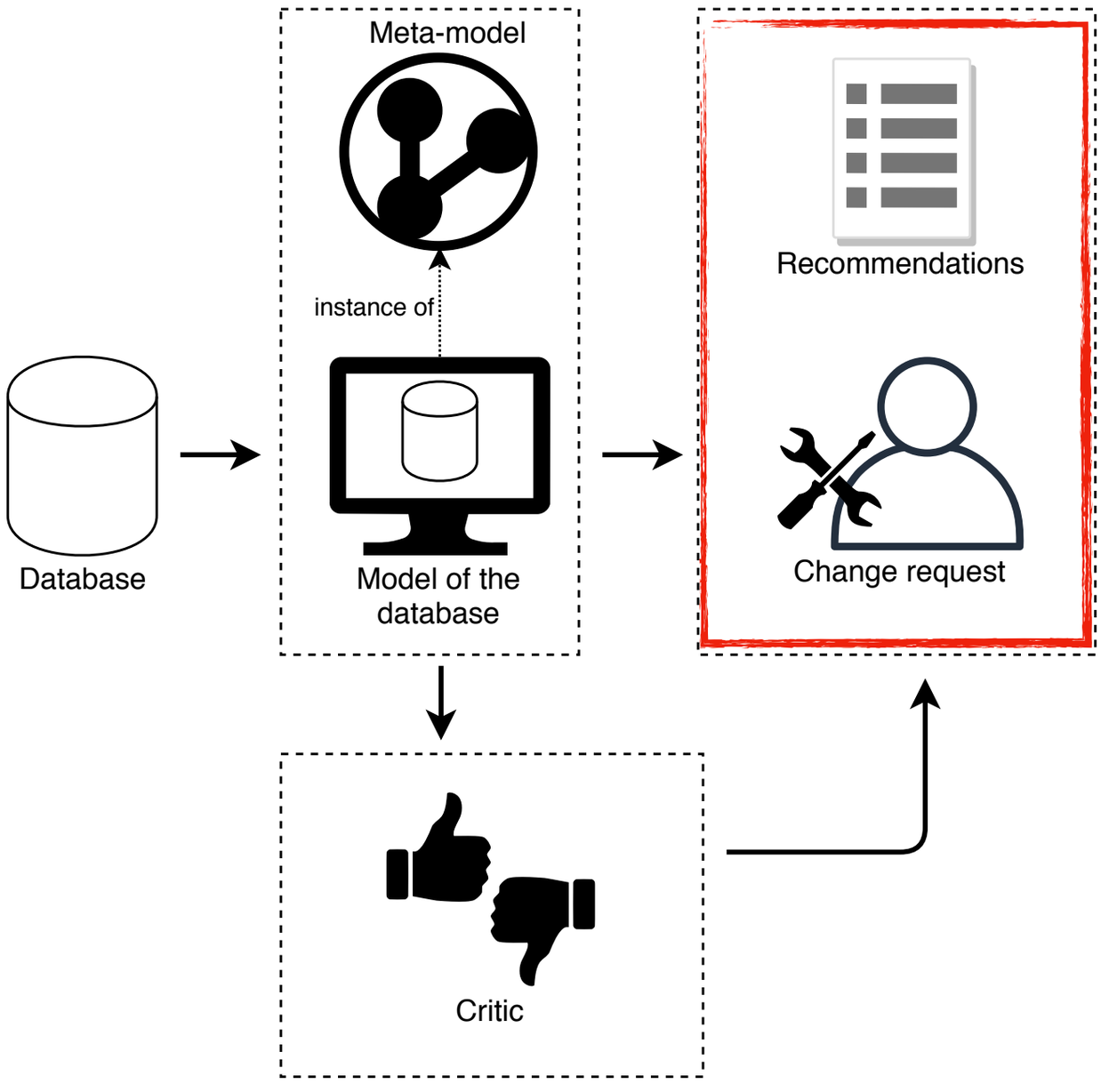
… and designed an approach to tackle it.



Including:

- a meta-model
- an approach to build it
- **a tool to assess DB quality**
- an approach to evolve DB
- empirical experiment(s) for each part

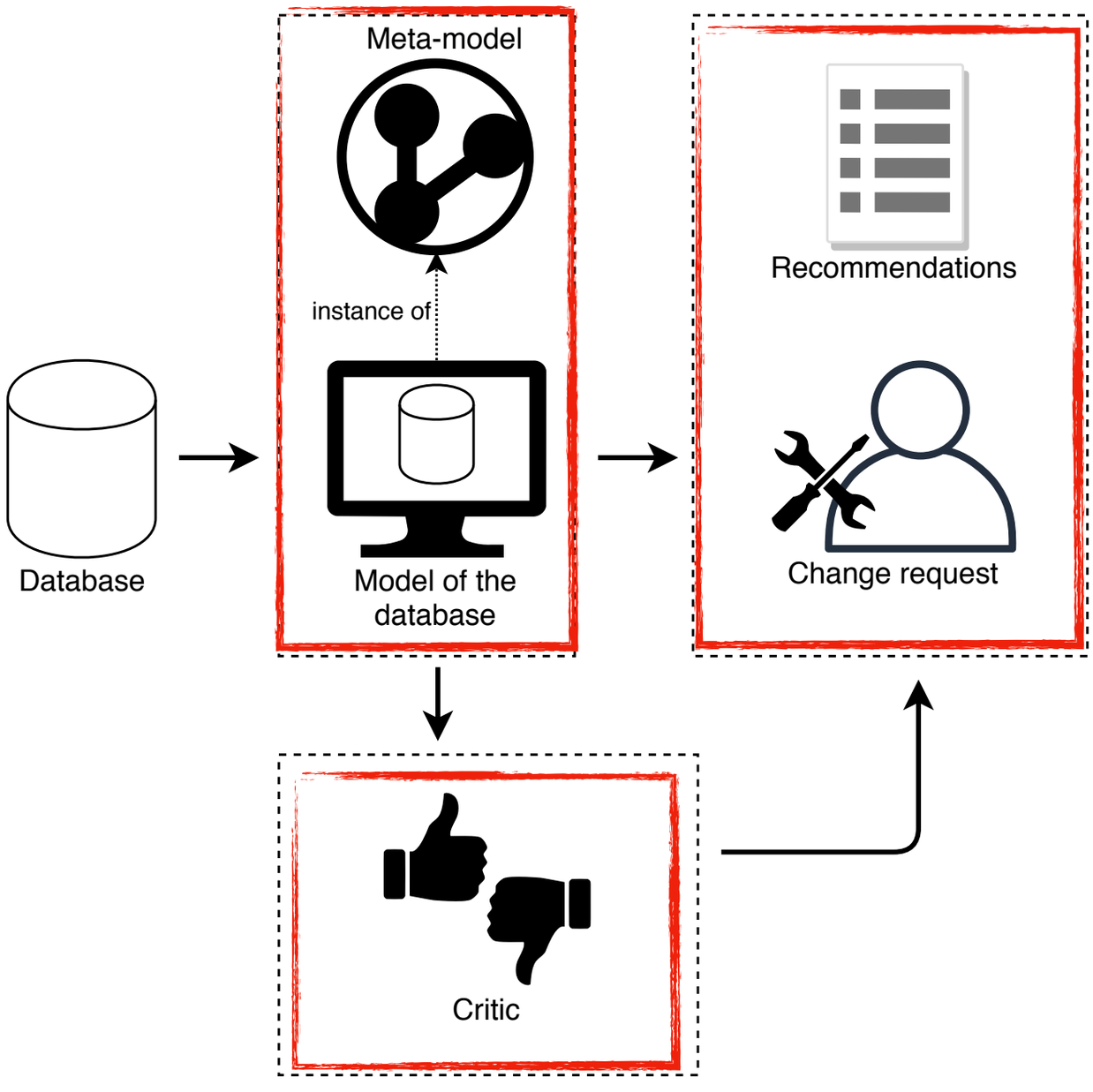
… and designed an approach to tackle it.



Including:

- a meta-model
- an approach to build it
- a tool to assess DB quality
- **an approach to evolve DB**
- empirical experiment(s) for each part

… and designed an approach to tackle it.



Including:

- a meta-model
- an approach to build it
- a tool to assess DB quality
- an approach to evolve DB
- **empirical experiment(s) for each part**

Publications

Conferences and Journals

1	CodeCritics Applied to Database Schema: Challenges and First Results (ERA track)	SANER'17
2	Relational Database Schema Evolution: An Industrial Case Study	ICSME'18
3	Rotten Green Tests	ICSE'19
4	Recommendations for Evolving Relational Databases	CAISE'20
5	Rotten Green Tests: A Replication Study (in submission)	ESE'20

Workshops

6	Software Engineering Issues in RDBMS, a Preliminary Survey	BENEVOL'17
7	Définition et identification des tables de nomenclatures	INFORSID'18
8	Software Engineering Techniques Applied to Relational Databases (doctoral track)	ASE'18
9	Rotten Green Tests, A first Analysis	IWST'18
10	Magic Literals in Pharo.	IWST'19

Publications

Conferences and Journals

1	CodeCritics Applied to Database Schema: Challenges and First Results (ERA track)	SANER'17
2	Relational Database Schema Evolution: An Industrial Case Study	ICSME'18
3	Rotten Green Tests	ICSE'19
4	Recommendations for Evolving Relational Databases	CAISE'20
5	Rotten Green Tests: A Replication Study (in submission)	ESE'20

Workshops

6	Software Engineering Issues in RDBMS, a Preliminary Survey	BENEVOL'17
7	Définition et identification des tables de nomenclatures	INFORSID'18
8	Software Engineering Techniques Applied to Relational Databases (doctoral track)	ASE'18
9	Rotten Green Tests, A first Analysis	IWST'18
10	Magic Literals in Pharo.	IWST'19

Included in the thesis

Publications

Conferences and Journals

1	CodeCritics Applied to Database Schema: Challenges and First Results (ERA track)	SANER'17
2	Relational Database Schema Evolution: An Industrial Case Study	ICSME'18
3	Rotten Green Tests	ICSE'19
4	Recommendations for Evolving Relational Databases	CAISE'20
5	Rotten Green Tests: A Replication Study (in submission)	ESE'20

Workshops

6	Software Engineering Issues in RDBMS, a Preliminary Survey	BENEVOL'17
7	Définition et identification des tables de nomenclatures	INFORSID'18
8	Software Engineering Techniques Applied to Relational Databases (doctoral track)	ASE'18
9	Rotten Green Tests, A first Analysis	IWST'18
10	Magic Literals in Pharo.	IWST'19

Additional publications

Future works

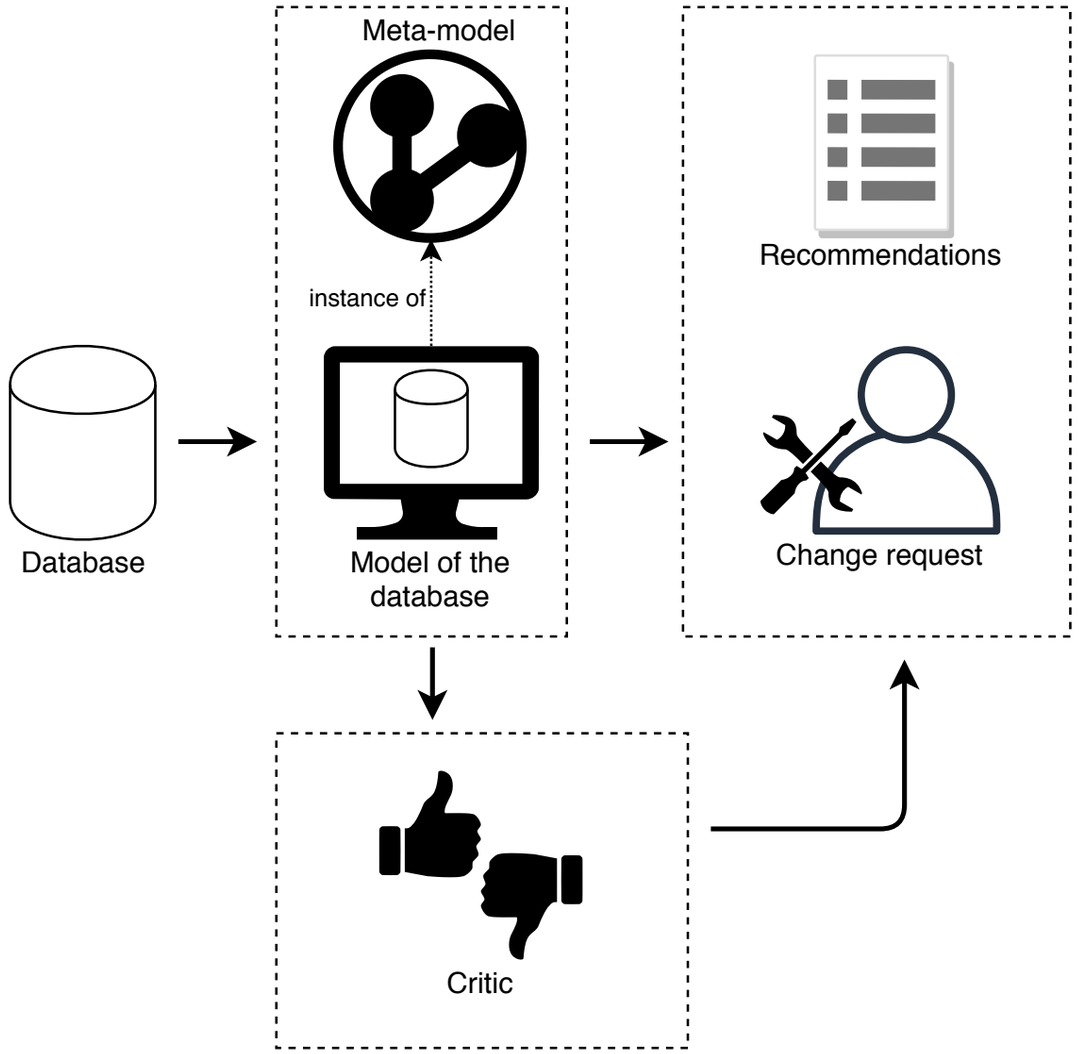
- Abstract syntax tree reification in the meta-model
- Support data transformation in evolution operators
- Simulate a sequence of operators on the model
- Conflict management when applying sequence of operators

Managing structural and behavioral evolution in relational database:

Application of Software Engineering techniques

Contributions in a nutshell:

- Identification of problems during database evolution
- Meta-model
- Quality assessment based on the meta-model
- Recommendations to evolve databases

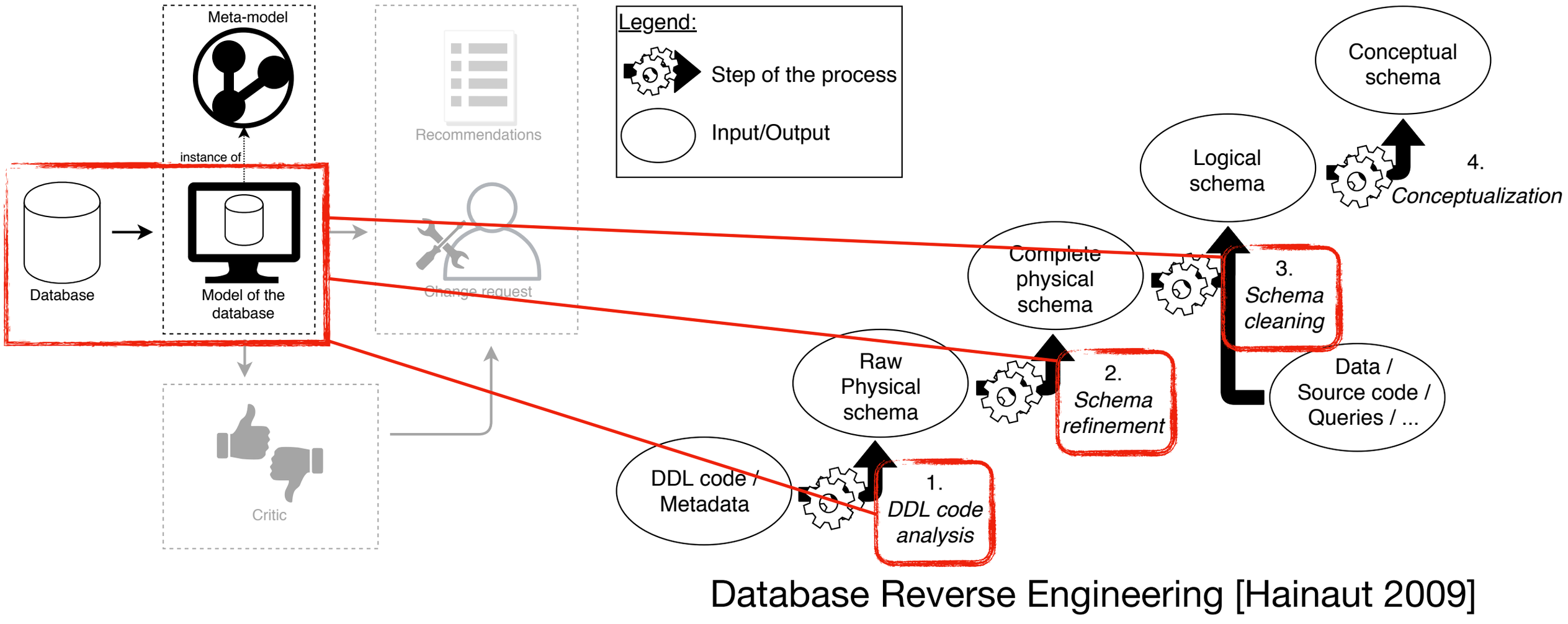


Relational Database Reverse Engineering Techniques

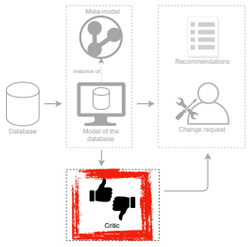
Techniques [Meurice 2017]**:**

- Database schema analysis
- Data analysis
- Graphical User Interface (GUI) analysis
- Static program analysis
- Dynamic program analysis

Instantiating the Meta-Model



Database Reverse Engineering [Hainaut 2009]



False positives

Three categories of violations can be distinguished:

1. Real design issues
2. Issues that the DBA accept to live with
3. Issues due to limitations of DBCritics

Classifying violations in these categories can not be automated.

On AppSI v10, the DBA analysed the 81 rule violations:
63% of the detected violations did point to quality problems

Can not be generalised but gives an idea.