Language Engineering for Language Migrations

Federico Tomassetti / Strumenta

LangDev CON 2024

Seville 17-19 October, 2024

https://langdevcon.org

Language Engineering for Legacy Migrations

The global market for legacy modernization was valued at **$15 billion** in 2022 and is expected to grow to **$24 billion** by 2026 (Modlogix).

Around **80% of organizations** state that outdated technology is hindering their ability to innovate, and 94% of executives say legacy systems severely limit their business agility (NTT Data).

MYJW

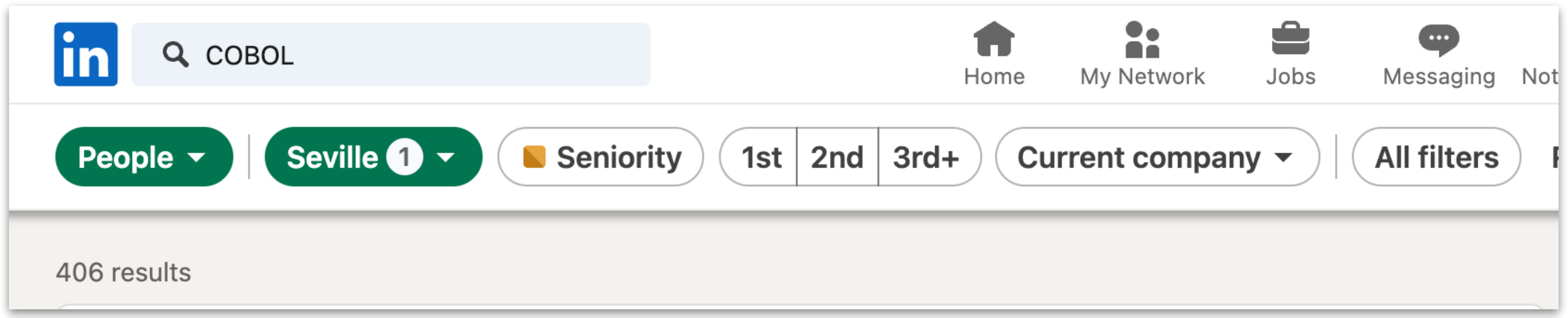
ProxyHands

Language Engineering for Legacy Migrations

Language	Usage	Typical Codebase Age	Estimated Codebase Size
COBOL	Business, finance, administrative systems (banking, insurance, government)	1960s–1980s	~200 billion lines
RPG	Manufacturing, IBM systems (AS/400, System/36)	1970s–1980s	Tens of millions of lines
FORTRAN	Scientific computing, engineering, simulations	1960s–1970s	Hundreds of millions of lines
Visual Basic (VB)	Enterprise desktop applications, automation (Windows)	1990s–2000s	Tens of millions of lines
PL/1	Data processing, business, scientific systems (IBM)	1960s–1980s	Millions of lines
Ada	Real-time systems, aerospace, defense	1980s–1990s	Millions of lines
CICS	Transaction processing (banking, airline reservations)	1970s–1980s	Millions to tens of millions of lines
Assembler	Low-level programming, hardware control	1950s–1970s	Millions of lines
4GLs	Database queries, business reporting, ERP	1980s–1990s	Tens to hundreds of millions of lines

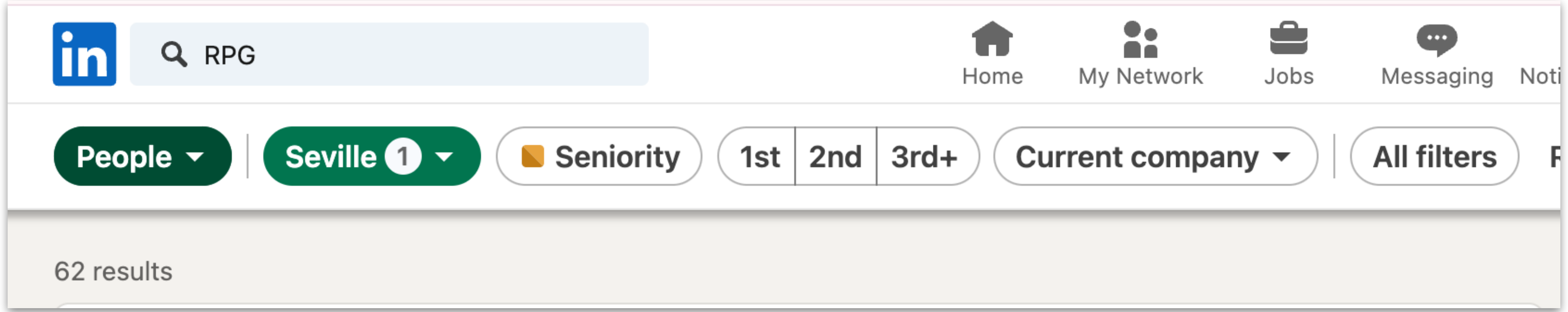
Problems with Legacy Code

Legacy Language Talent Pool:
- COBOL: ~2 million developers globally (majority nearing retirement)
- RPG: Developer pool is shrinking, with very few new developers entering the field
- FORTRAN, PL/1, Assembly: Limited availability of specialists



COBOL

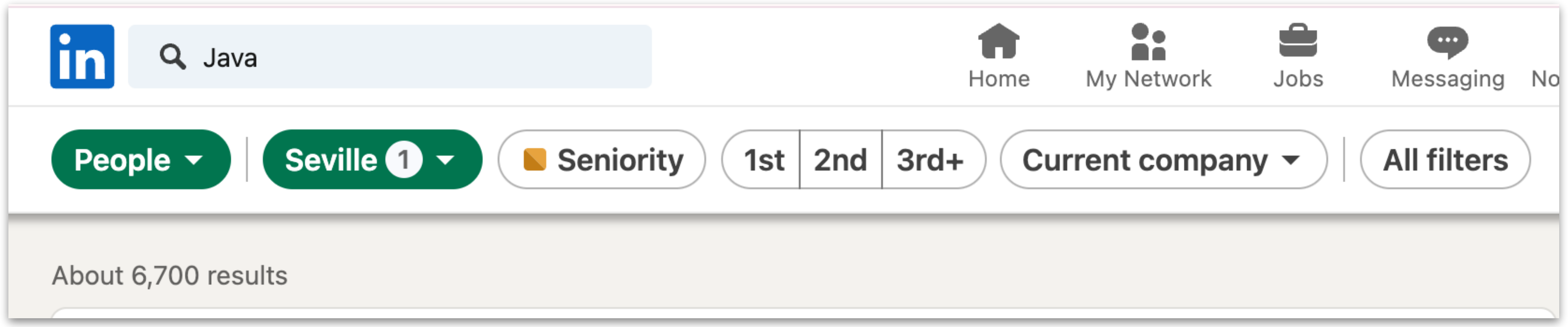
406 results



RPG

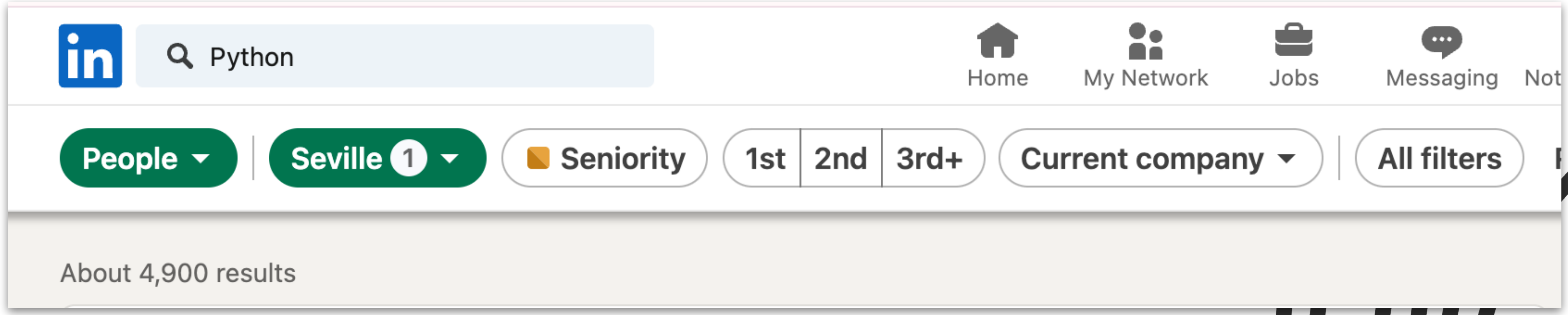
62 results

Modern Languages Talent Pool:
- Python: ~8 million developers globally
- Java: ~7 million developers globally
- Kotlin, JavaScript: Growing at a rapid pace



Java

About 6,700 results



Python

About 4,900 results

Problems with Legacy Code

“ Language Engineering for Legacy Migrations

Do not blame who has Legacy Code.

Manual Rewrites: what can possibly go wrong?

They fail

Some reports suggest that failure rates for large IT projects, such as system rewrites, can range from **50% to 70%**. This is due to factors like scope creep, lack of familiarity with legacy code, and the complexity of accurately reproducing the system's functionality (GenU - GenUI).

They cost

The cost of a manual rewrite is highly variable but can range from **$6 to $23 per line of code** depending on the complexity and specific requirements of the system. For large legacy systems with millions of lines of code, this can easily push the total cost into the millions of dollars (GenU - GenUI) (RTS Labs).

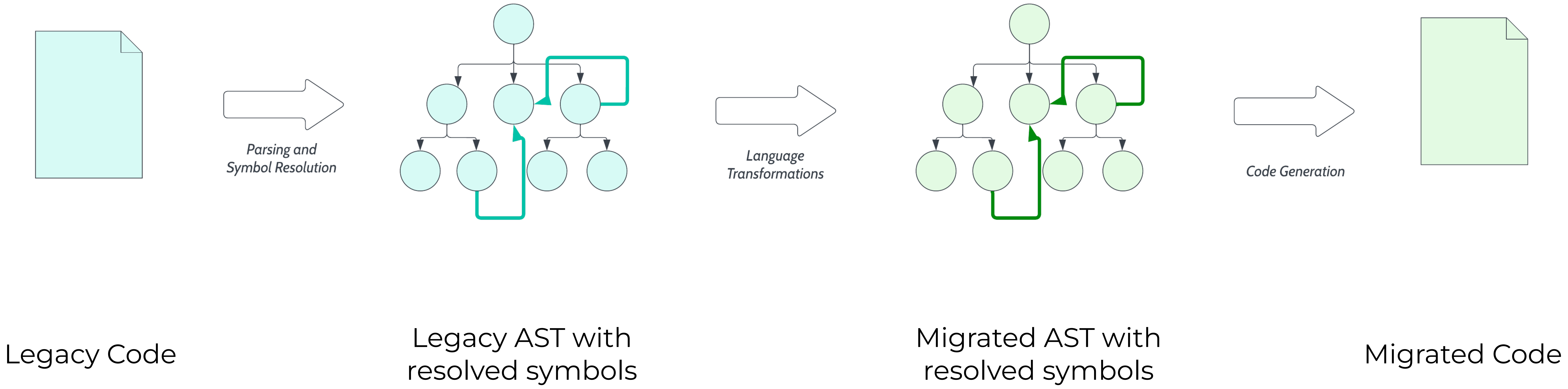
Operational costs can also increase dramatically during the rewrite process, especially if both the old and new systems need to run in parallel for some time (YTG Services).

They take forever

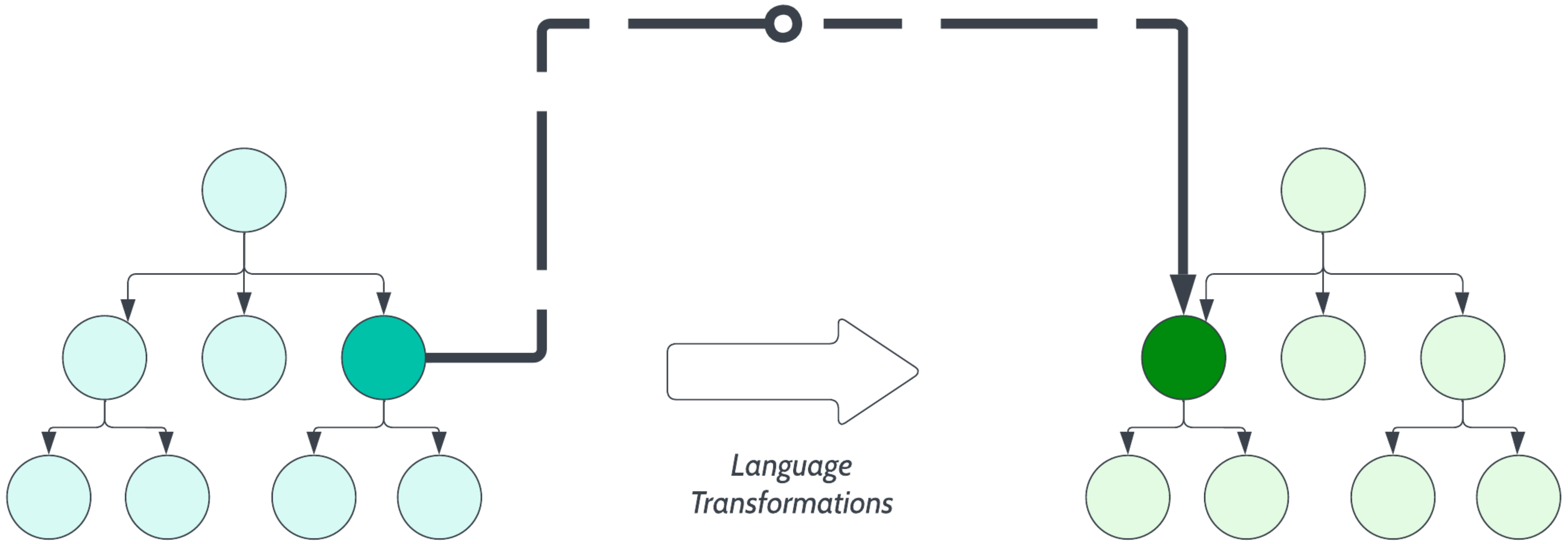
A manual rewrite can take **years** to complete. The timeline depends on the size of the system, but for large-scale systems, it's common for rewrites to span multiple years. During this time, the organization also faces the risk of disruption, increased costs, and delayed project completion (YTG Services) (RTS Labs).

Code Insight Studio Demo

Migrations' Architecture



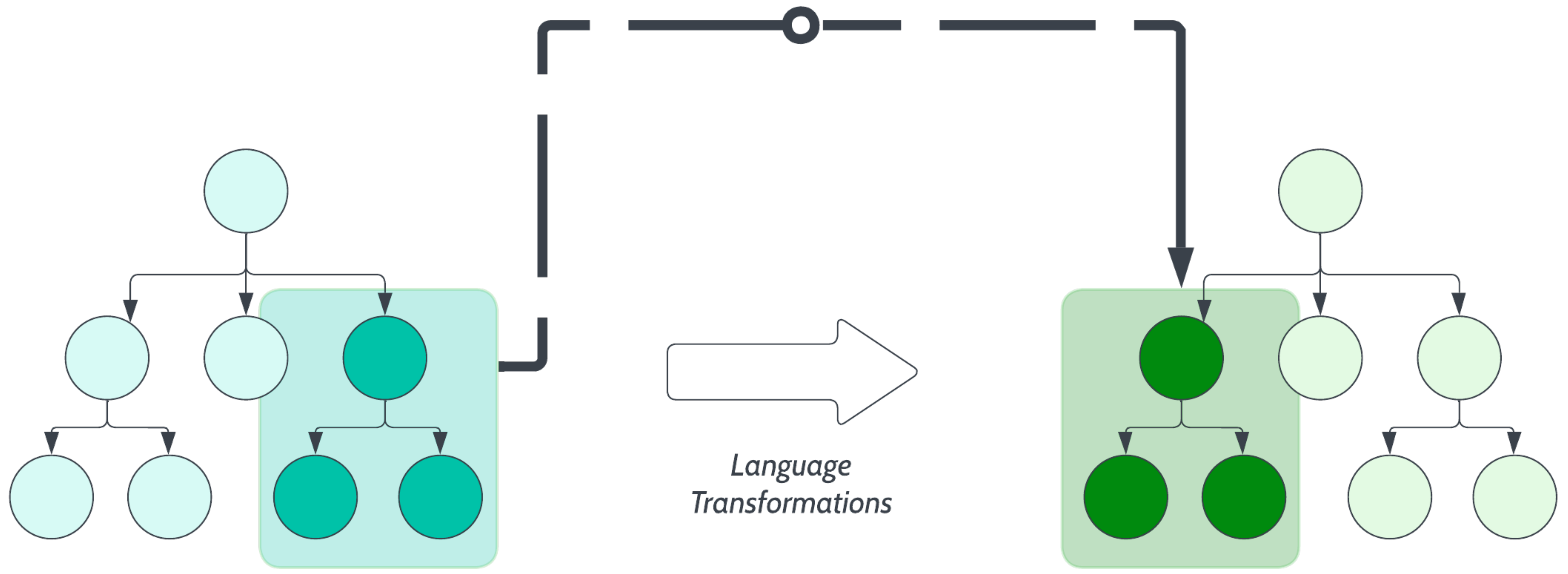
Migrations’ Architecture



```
d NoKeys                 s              1
```

```
String noKeys;
```

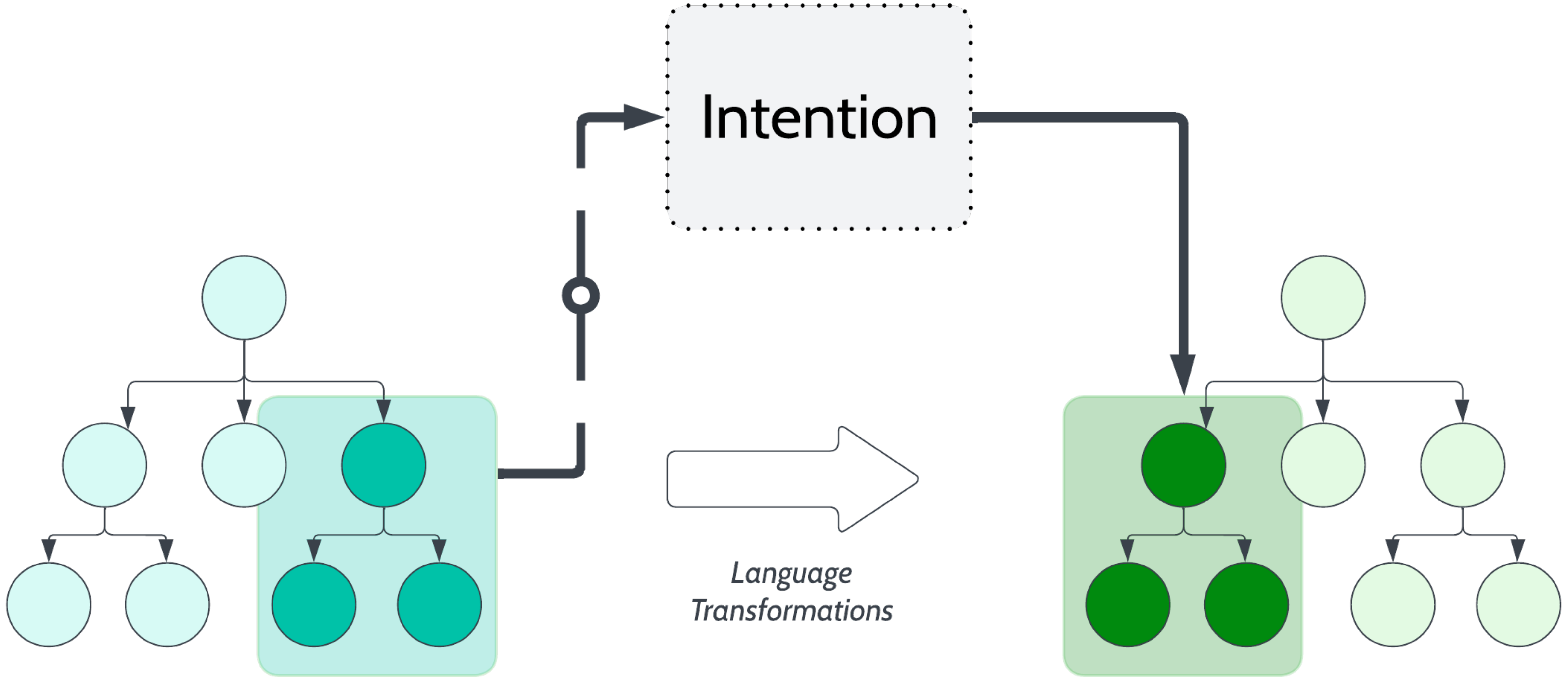
Migrations' Architecture



```
C    READ   FILE
C    DOW    NOT %EOF(FILE)
C    DELETE FILE
C    READ   FILE
C    ENDDO
```

```
String sql = "DELETE FROM tableName";
Statement stmt = connection.createStatement();
stmt.executeUpdate(sql);
```

Migrations' Architecture



```
C    READ   FILE
C    DOW    NOT %EOF(FILE)
C    DELETE FILE
C    READ   FILE
C    ENDDO
```

```
String sql = "DELETE FROM tableName";
Statement stmt = connection.createStatement();
stmt.executeUpdate(sql);
```

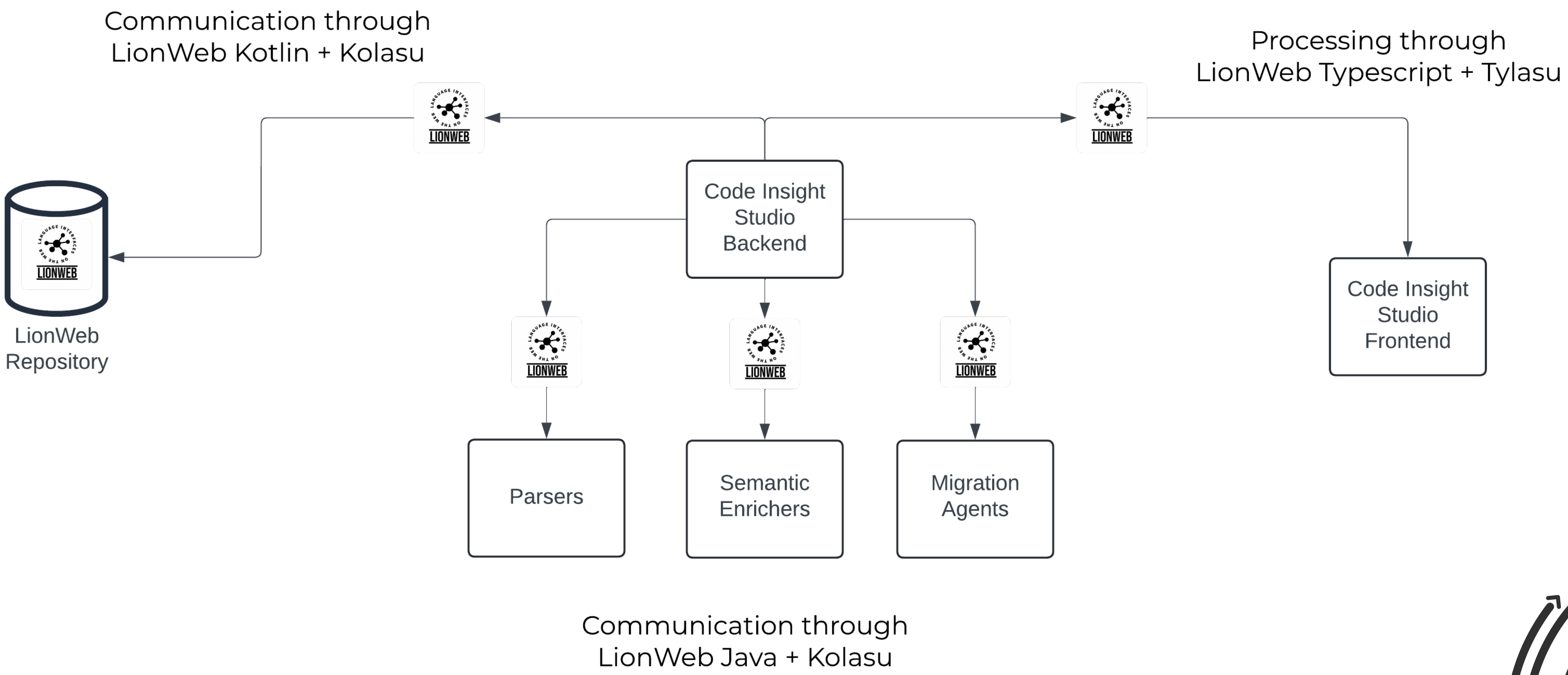
Why these two-levels approach?

- The more we cover with patterns, the more idiomatic it is
- The construct-to-construct is a fall-back

Customization of Migrations

- Pick the target language
- Pick the target framework
- Pick supporting libraries
- Pick code style
- Identify codebase-specific patterns
- Discuss the transformations (pattern-level and construct-level)

The Role of LionWeb



Communication through LionWeb Kotlin + Kolasu
Processing through LionWeb Typescript + Tylasu
LionWeb Repository
Code Insight Studio Backend
Code Insight Studio Frontend
Parsers
Semantic Enrichers
Migration Agents
Communication through LionWeb Java + Kolasu

Challenges in Legacy Migrations

- Ensuring behavioral parity between old and new systems
- Knowledge transfer between teams
- Removal of clones in the migration
- Removal of dead-code
- Documentation of the migrated system

Final Considerations

94% of executives say legacy systems severely limit their business agility

This problem seems a good fit, for our mission

Better Tools
for
Better Work

Q&A

Thanks

MYJW

ProxyHands