Applying Social Network Analysis Techniques to Architectural Smell Prediction

Dr Antonela Tommasel

ISISTAN Instituto Superior de Ingenieria de Software Tandil
CONICET
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Software Evolution & Dependencies

As software systems evolve, the **amount and complexity of the interactions** amongst their components increases.

- More coupling.
- **"*Undesired*" dependencies** amongst certain components.
- Degradation of intended design → Architectural Smells appear!

Early detection is important to plan ahead for actions to stop degradation

- We do not only need to fix local problems.
- Know which modules are likely to get coupled in the future.

Anticipate problems! **Proactively look for solutions.**

Example: Cycle Evolution

What do we want?

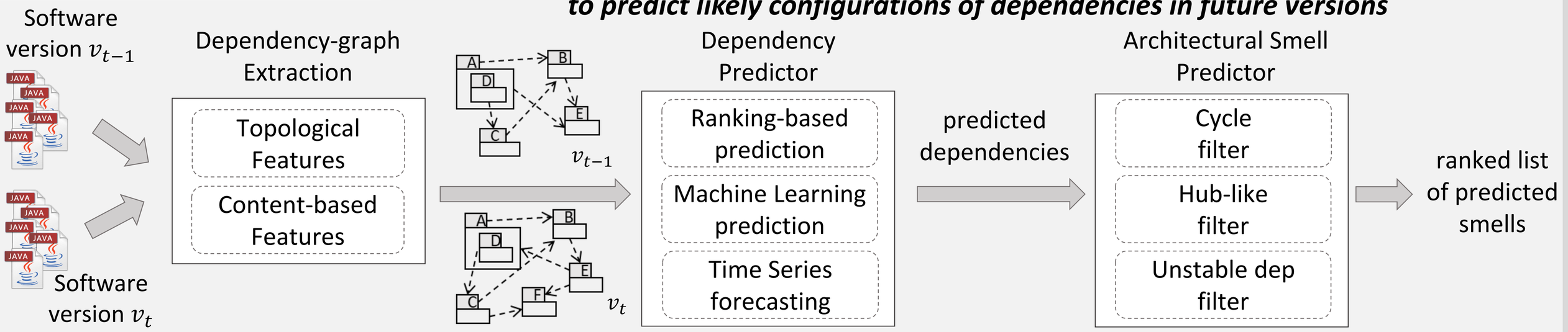
Predict when a dependency-related problem is likely to manifest!

- Software systems and underlying architecture behave as social networks.
- Changes can be predicted based on the appearance of dependencies between design elements.

We argue that **Social Network Analysis** techniques can be used for **software dependency prediction**!

Smell Prediction Proposal

To what extent we can leverage on information from previous software versions to predict likely configurations of dependencies in future versions

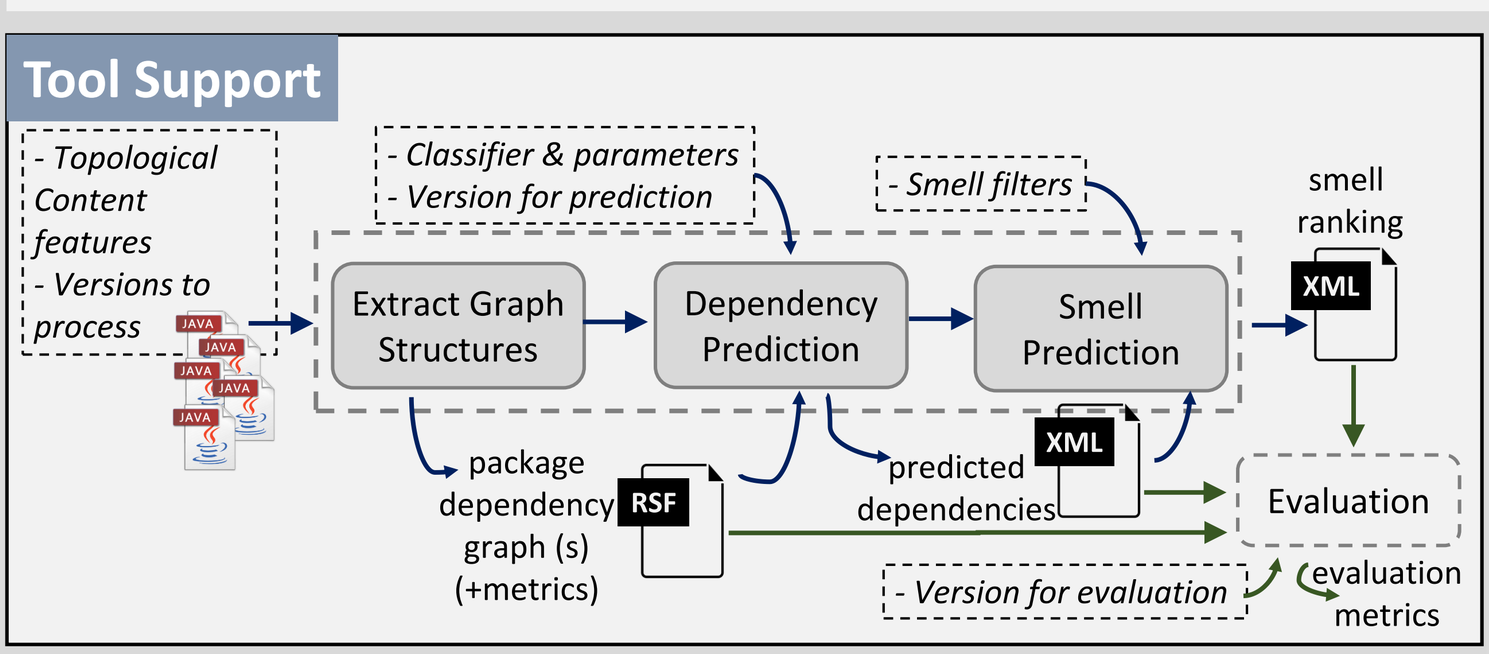


- Build a graph $DG(V, E)$ for system version n, where:
  - Each node v in V is Java package, and each edge *e* in *E* is a usage relationship between a pair of packages v_1 and v_2.
  - Edges represent similarity between packages (topological or content-based).

Dependencies predicted based on…		
	current version	*Link prediction* Homophily-based
	previous and current version	Machine Learning approach Classifier-based
	history of versions	Time series Forecasting Dynamic SNA

- The prediction of a dependency is not enough to predict the appearance of an architectural smell.
  - **Not every predicted dependency might cause an smell to emerge.**
- Predicted dependencies undergo a filtering process.
  - Filters are **smell-dependent**.

Tool Support



There is still work to do!

- More features. Design metrics? OO metrics?
- Analyse other dependency-based problems!
- Can we predict the appearance of new nodes (e.g. new packages, classes)?
- Can we predict the disappearance of dependencies?

Lessons Learned

Machine Learning techniques have the potential for Link Prediction applied to software dependencies

- An initial evaluation with cycles & hubs showed a good performance!
  - High recall, low precision.
- Including content-based features improves dependency prediction.
- **Leveraging on information from previous versions gives reasonable predictions**, although not all versions seem useful.
- The choice of the filter variant (for a given smell type) can affect both recall and precision.
  - We preferred good recall over precision in the analysed cases.