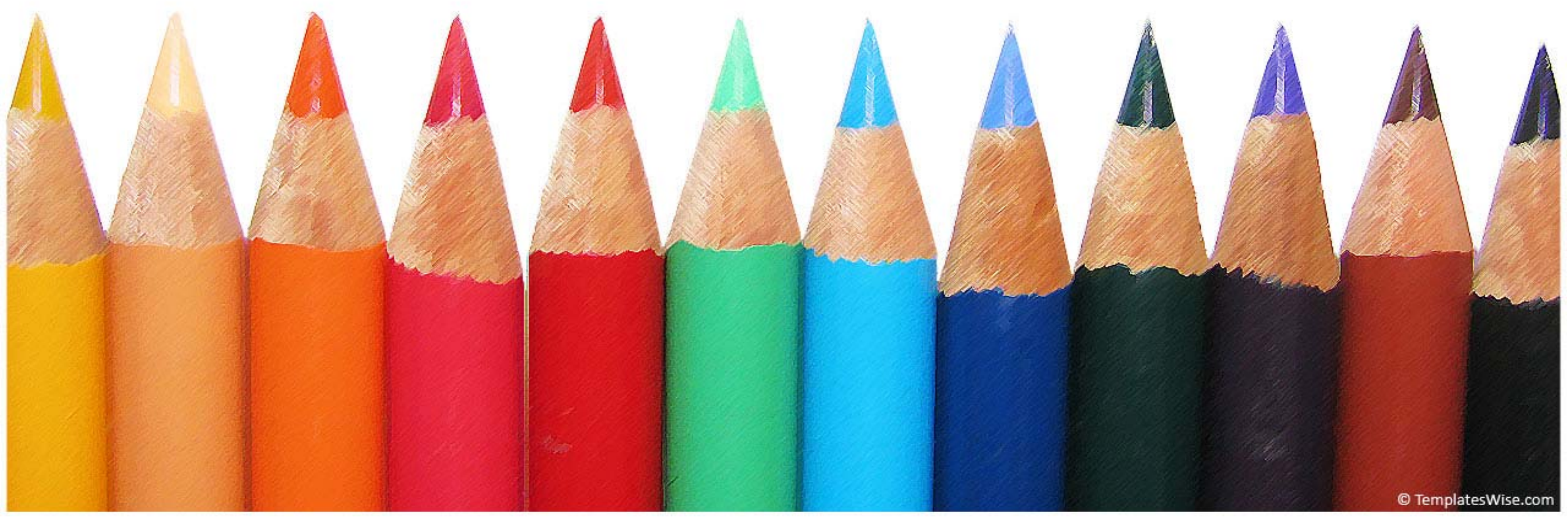
Using Formal Concept Analysis to Construct and Visualise Hierarchies of Socio-Technical Relations

Michel Wermelinger, Yijun Yu
The Open University, UK
Markus Strohmaier
Graz University of Technology, and Know-Center, Austria



1. Motivation

- Software engineering is socio-technical activity
  - Global and open source software development led to increased interest in and relevance of social aspects
- Need for representing socio-technical relations
- Bipartite graphs of software artefacts and people
  - Ad-hoc arc semantics, depending on relation
  - Ad-hoc flat layout, often hard to read
  - Relevant relations lost among many nodes and arcs
- Sought improvements:
  - More compact, intuitive, and explicit representation
  - Distinguish 'hierarchical' importance of artefacts, people and their relations.

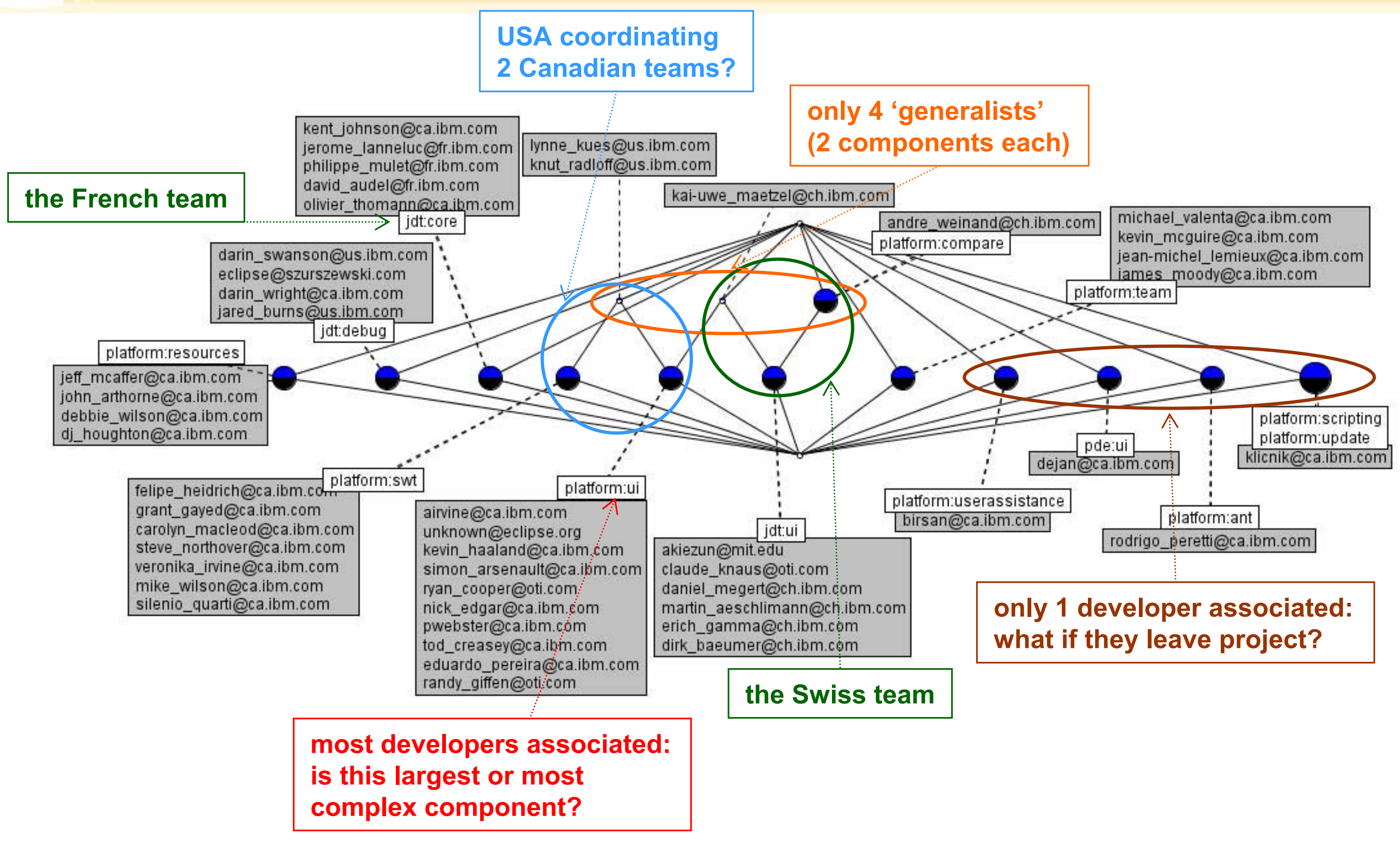
2. General Approach

- Obtain a bipartite socio-technical network
- Compute socio-technical concept lattice
  - Apply formal concept analysis (FCA) theory
  - Use free tool ConExp (Concept Explorer)
  - Input: bi-partite network
  - Output: concept lattice (1 node per concept)
  - A concept clusters *all* artefacts associated to the *same* people
  - Hierarchy is partial ordering of clusters (arc semantics: subset)
- Visualise hierarchy interactively using ConExp
- Study different and evolving socio-technical relations
  - Repeat 1.-3. for various relations and system releases

3. Example

- Eclipse IDE
  - Has non-trivial social and technical structure
  - IBM lead allows social continuity to be traced
  - Bugzilla repository available
- Bi-partite socio-technical networks
  - Nodes denote people *p* and Eclipse components *c*
  - Arc *p*→*c* if *p* associated to *k* or more bug reports for *c*, for a given *k*
  - 3 associations with bug reports: reporter, assignee or discussant
- Socio-technical concept lattice
  - Usually, person at level *n* from bottom is associated to *n* components
  - Hence 'specialists' at bottom, 'generalists' at top of lattice
  - Each node includes all its ancestors' people and all its descendants' components

4. Eclipse 1.0, assignees, k = 10



USA coordinating 2 Canadian teams?

only 4 'generalists' (2 components each)

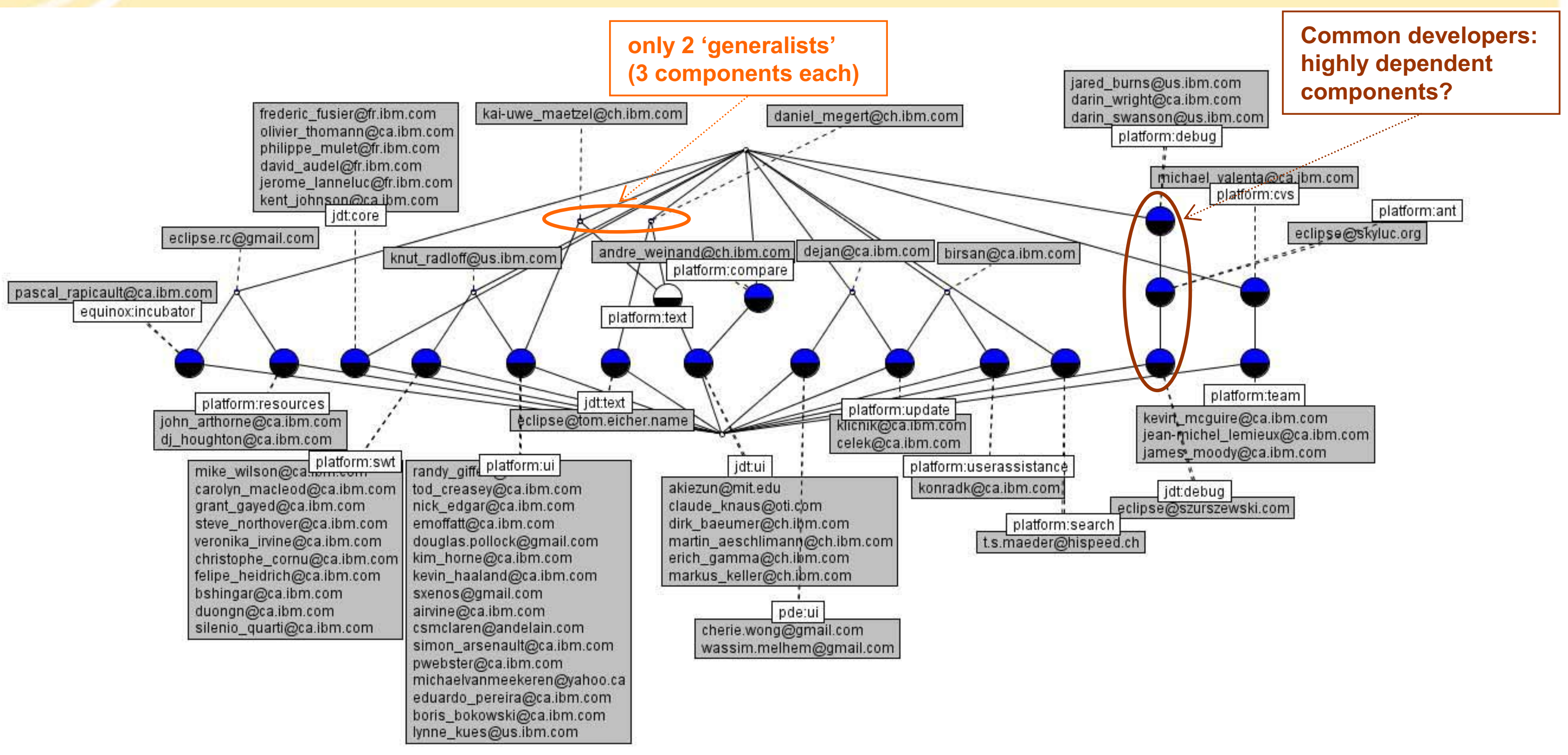
the French team

the Swiss team

most developers associated: is this largest or most complex component?

only 1 developer associated: what if they leave project?

5. Eclipse 3.0, assignees, k = 100

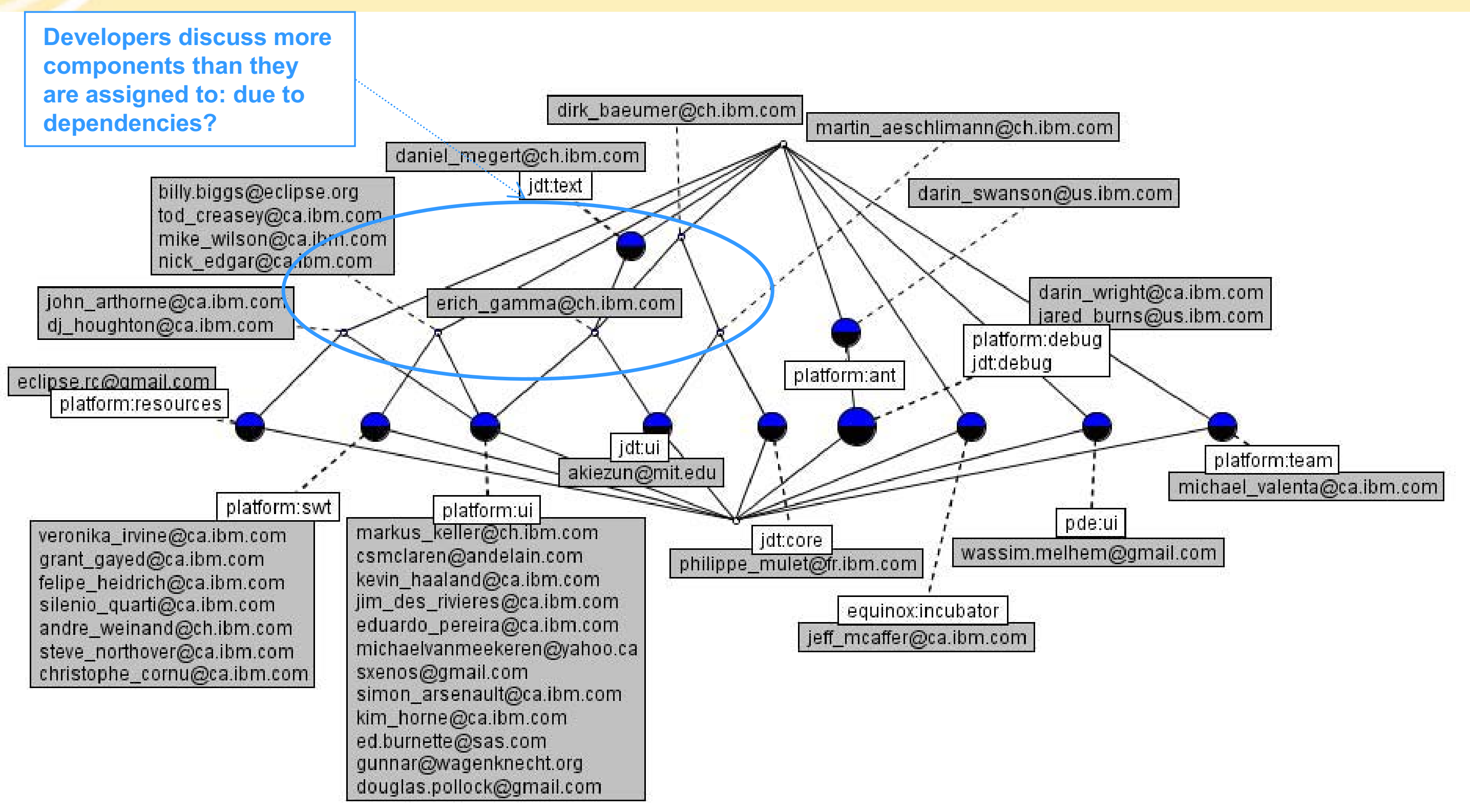


only 2 'generalists' (3 components each)

Common developers: highly dependent components?

- Used higher *k* because bug reports accumulate over time
- Geographical and workload distribution like release 1.0

6. Eclipse 3.0, discussants, *k* = 100



Developers discuss more components than they are assigned to: due to dependencies?

- Fewer people and components than in assignees lattice
  - Developers don't discuss all reports they are assigned to

7. Conclusions

- Novel application of Formal Concept Analysis
  - Clustering and ordering of socio-technical relations
  - General tool-supported approach
- Some advantages over bi-partite graphs
  - More scalable: not one node per person and artefact
  - More explicit: related people & artefacts in same node
  - More intuitive: uniform vertical layout & arc semantics
- Helps spot expertise and potential problems
  - Generalist and specialist people
  - Artefacts with too many or too few people associated
  - Undesired or absent communication/coordination

For more details, see our paper in Proc. ICSE'09 (companion volume), pp. 327-330.