Exploiting Traceability Uncertainty between Architectural Models and Code

Achraf Ghabi & **Alexander Egyed**

achraf.ghabi@jku.at & alexander.egyed@jku.at

WICSA/ECSA 2012

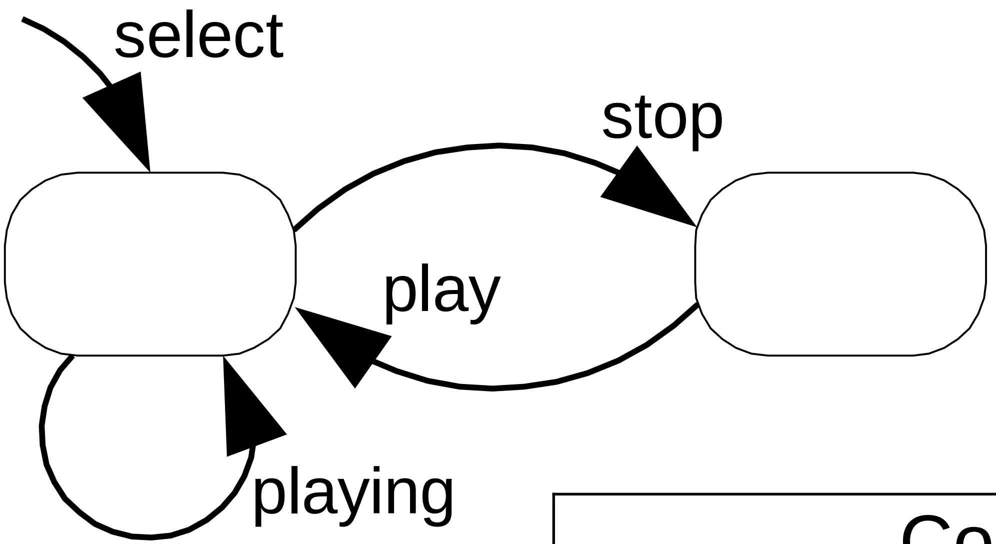
SEA *Institute for Systems Engineering and Automation*

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Traceability

- **Informal definition:**
  The collection of traces which are documenting the relationship between two artifacts.
- Useful particularly at maintenance time
- In this work:
  - Two Artifacts: Model (solution elements) and Code (code elements)
  - Relationship: „is implemented by“
- Trace example:
  - Model element A is implemented by code element C

Model and Code: VoD Client



Code: classes
MPEGDecoder.Button
MPEGDecoder.DStore
MPEGDecoder.LFrame
MPEGDecoder.Movie
MPEGDecoder.Header

Trace Matrix: Certainty

Code\mode elements	play	playing	select	stop
MPEGDecoder.Button			X	
MPEGDecoder.DStore	X	X	X	X
MPEGDecoder.LFrame		X		X
MPEGDecoder.Movie			X	
MPEGDecoder.Header		X		

Trace	X
No-Trace	

Uncertainty In Practice

Traceability is difficult:

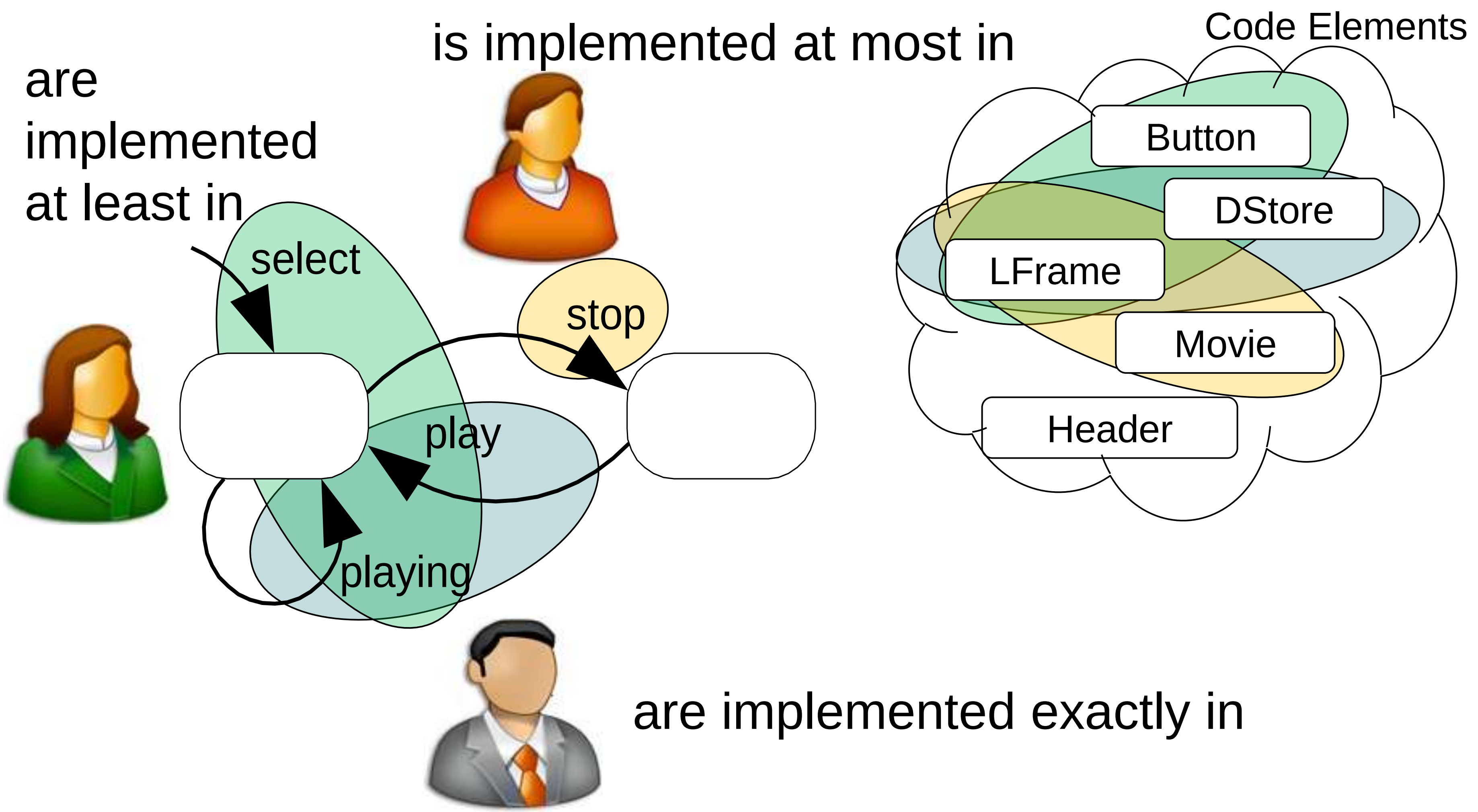
- Loosing some key developers
- Understanding of the entire code
- Forgotten details
- Incorrect recollection of facts
- Changes in model/code

→ Engineers might be uncertain about some trace relationships.

Goal

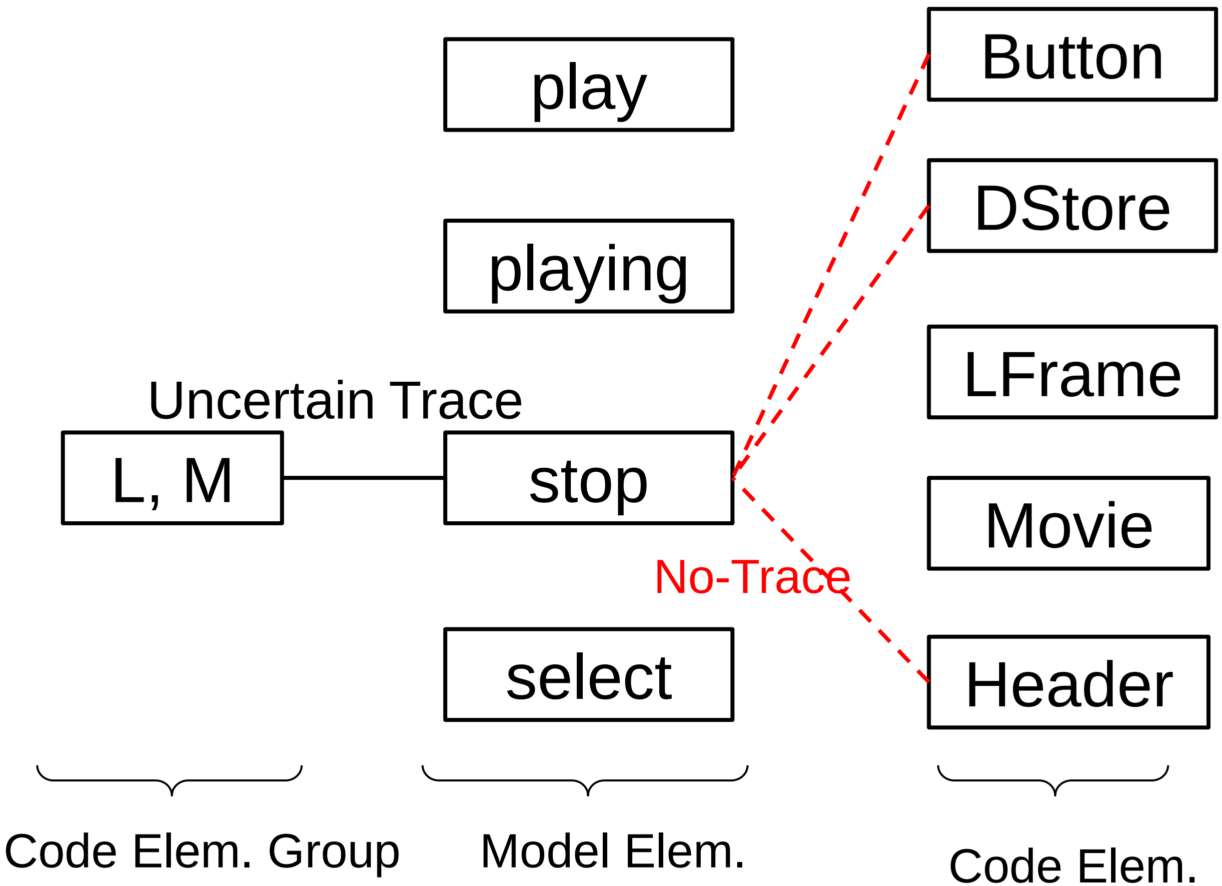
- Allow the engineer to express what she REALLY knows about a system
- Help engineer to
  - Detect incorrectness/inconsistency in her knowledge
  - Derive further traceability information based on her knowledge

Expressing Uncertainty



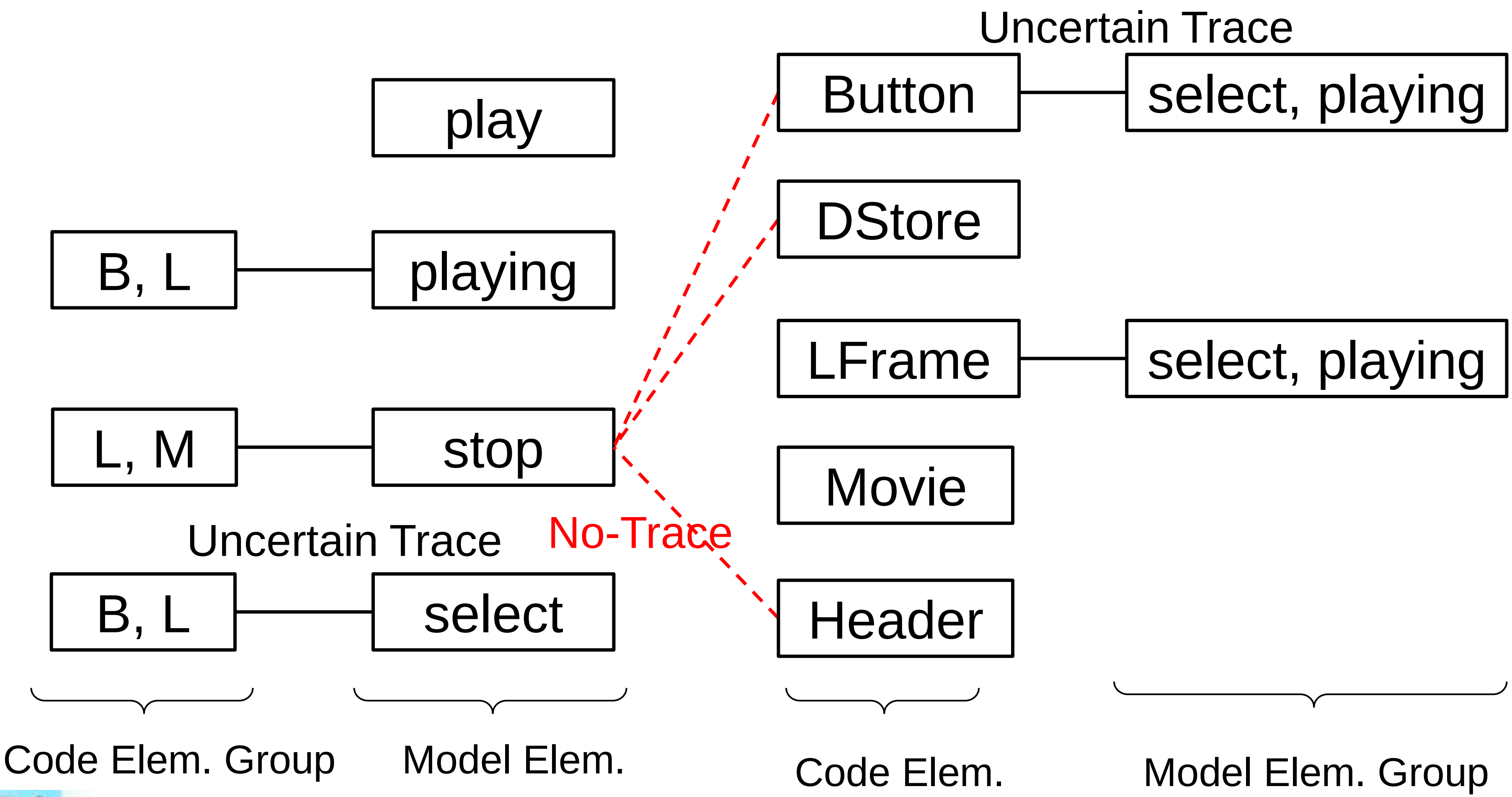
Footprint Graph

`{stop} implAtMost {LFrame, Movie}`



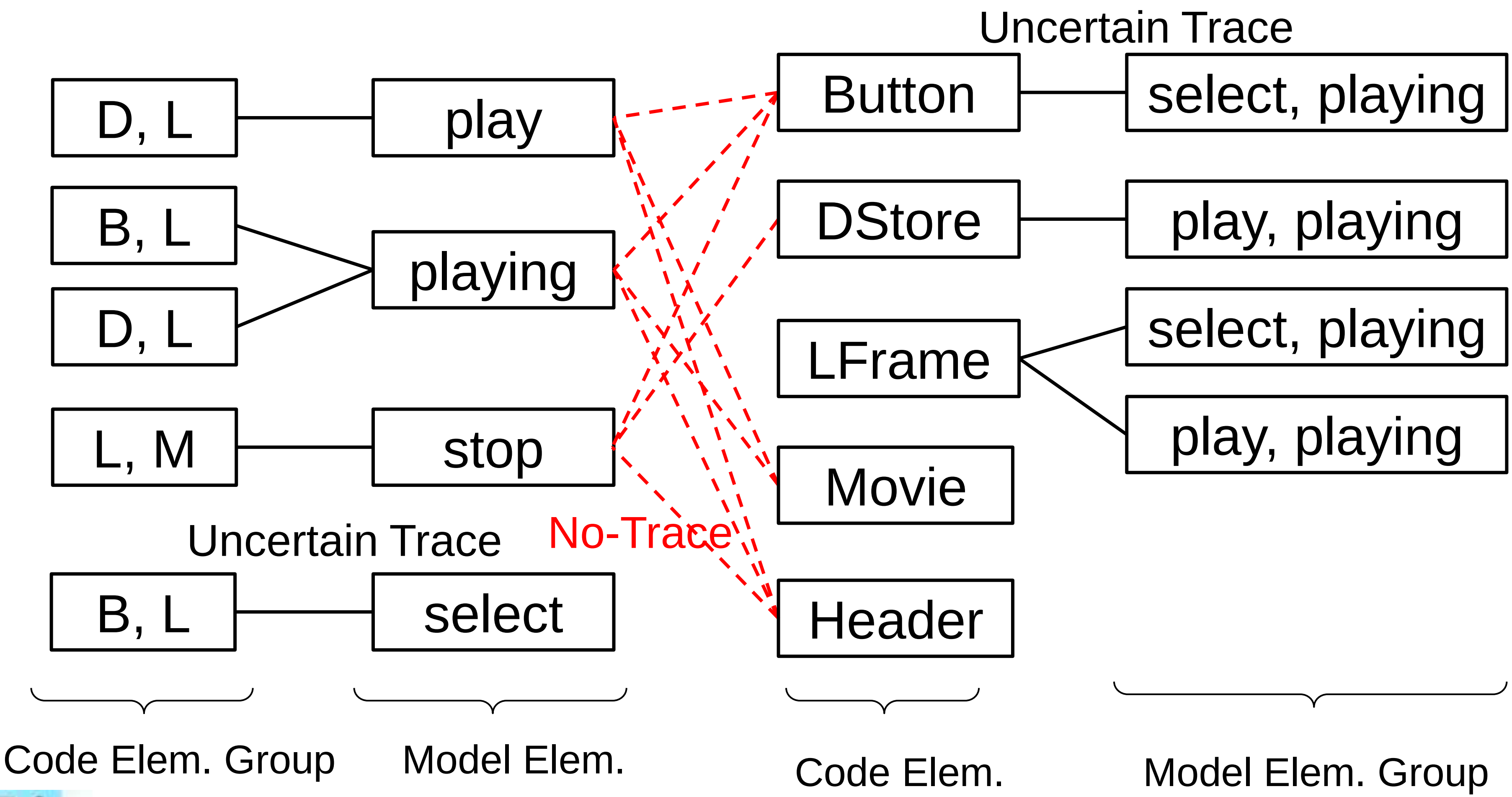
Footprint Graph

`{select, playing}` *`implAtLeast`* `{Button, LFrame}`

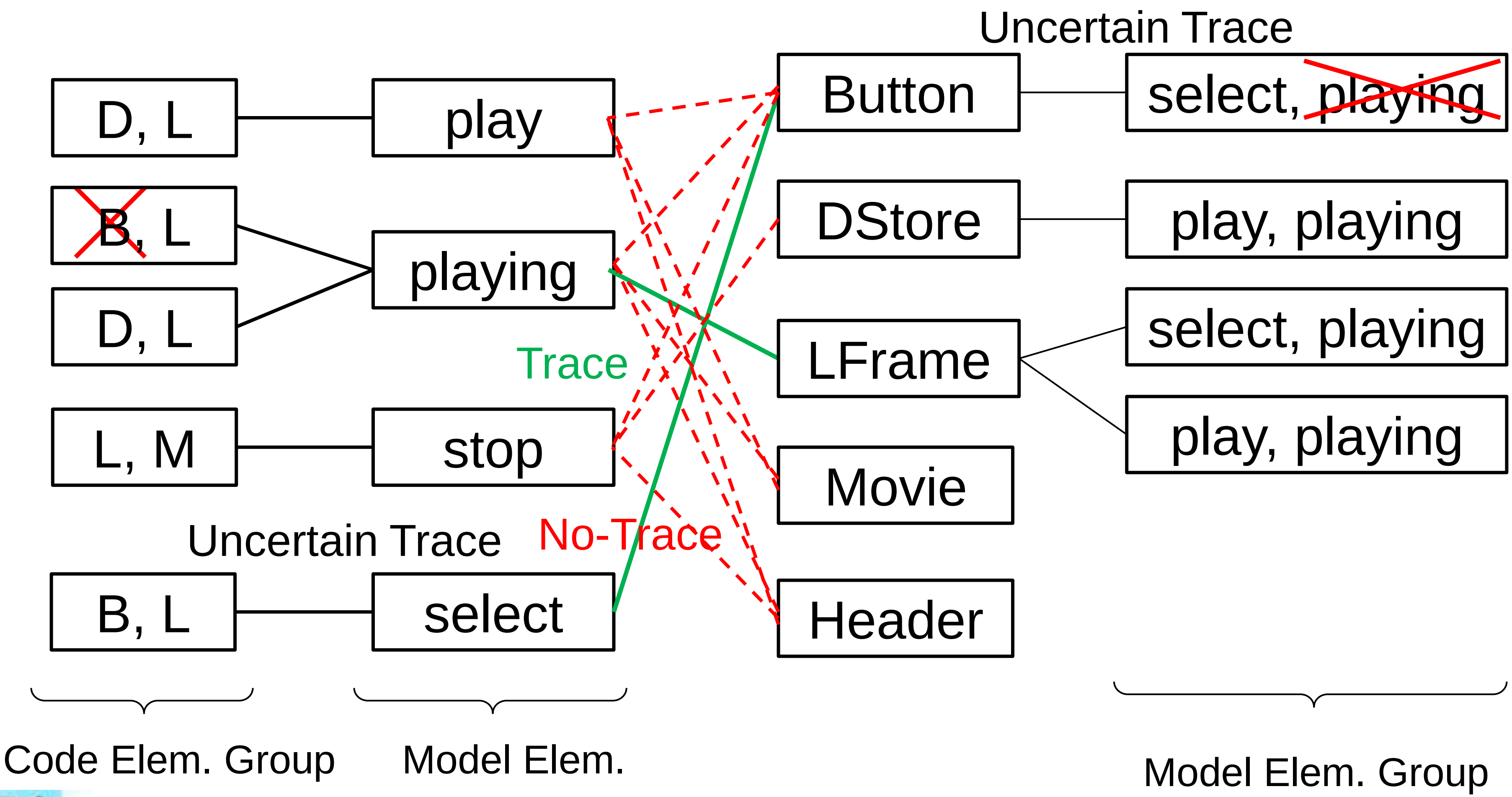


Footprint Graph

{play, playing} *implExactly* {DStore, LFrame}



Propagation Rules



Result in Trace Matrix

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- Filled the TM using uncertainties
- But not complete
- Correctness depends on the developer's knowledge.

Code	play	playing	select	stop
Button			X	
DStore				
LFrame		X		
Movie				
Header				

Trace	X
No-Trace	
Group	

Further Input

- In big systems usually multiple developers will do the traceability.

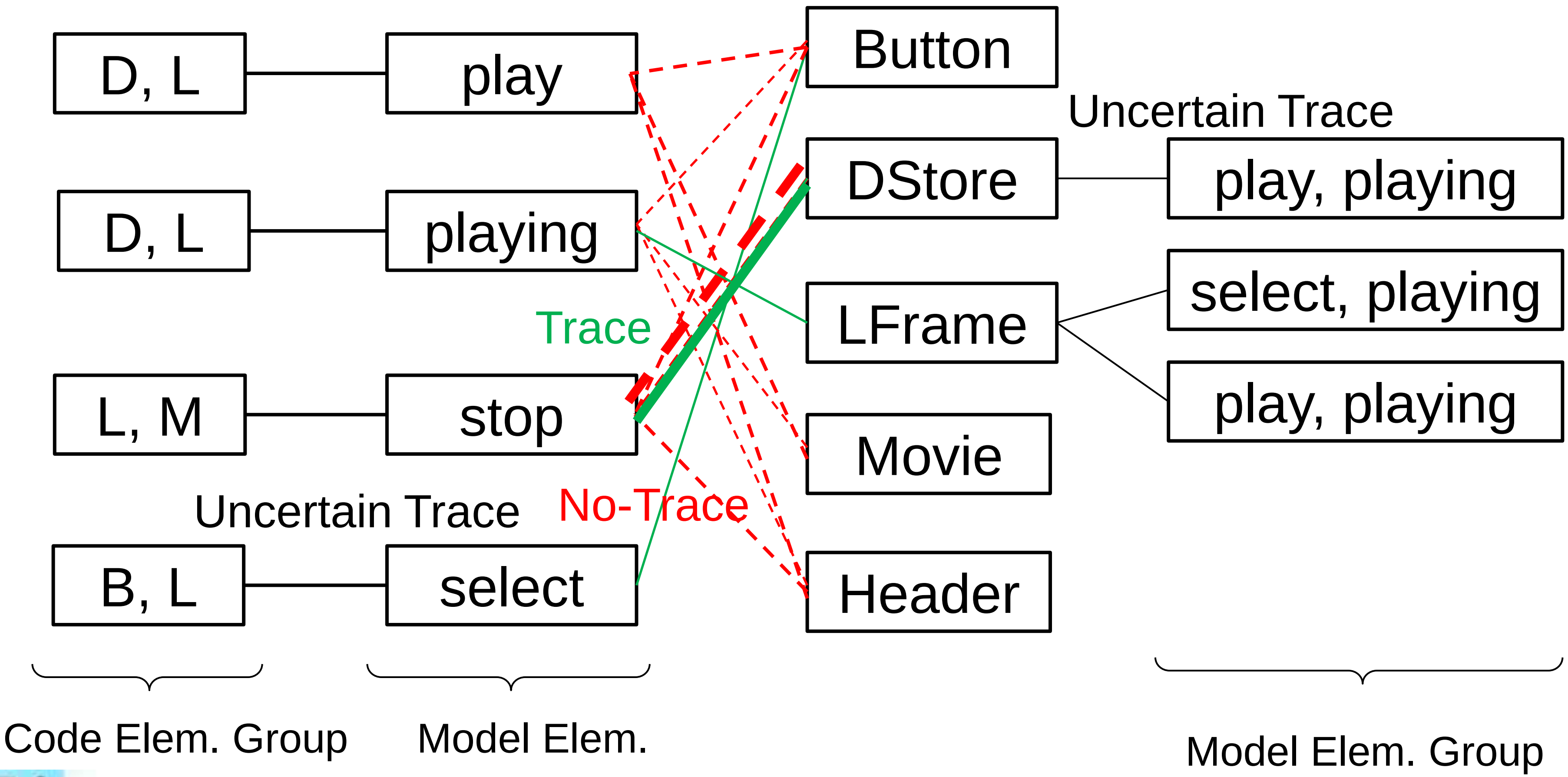
→ Different understanding of the system.

- Another Engineer introduces this input in her description:

`{stop}` *`implAtLeast`* `{DStore}`

Knowledge Conflict

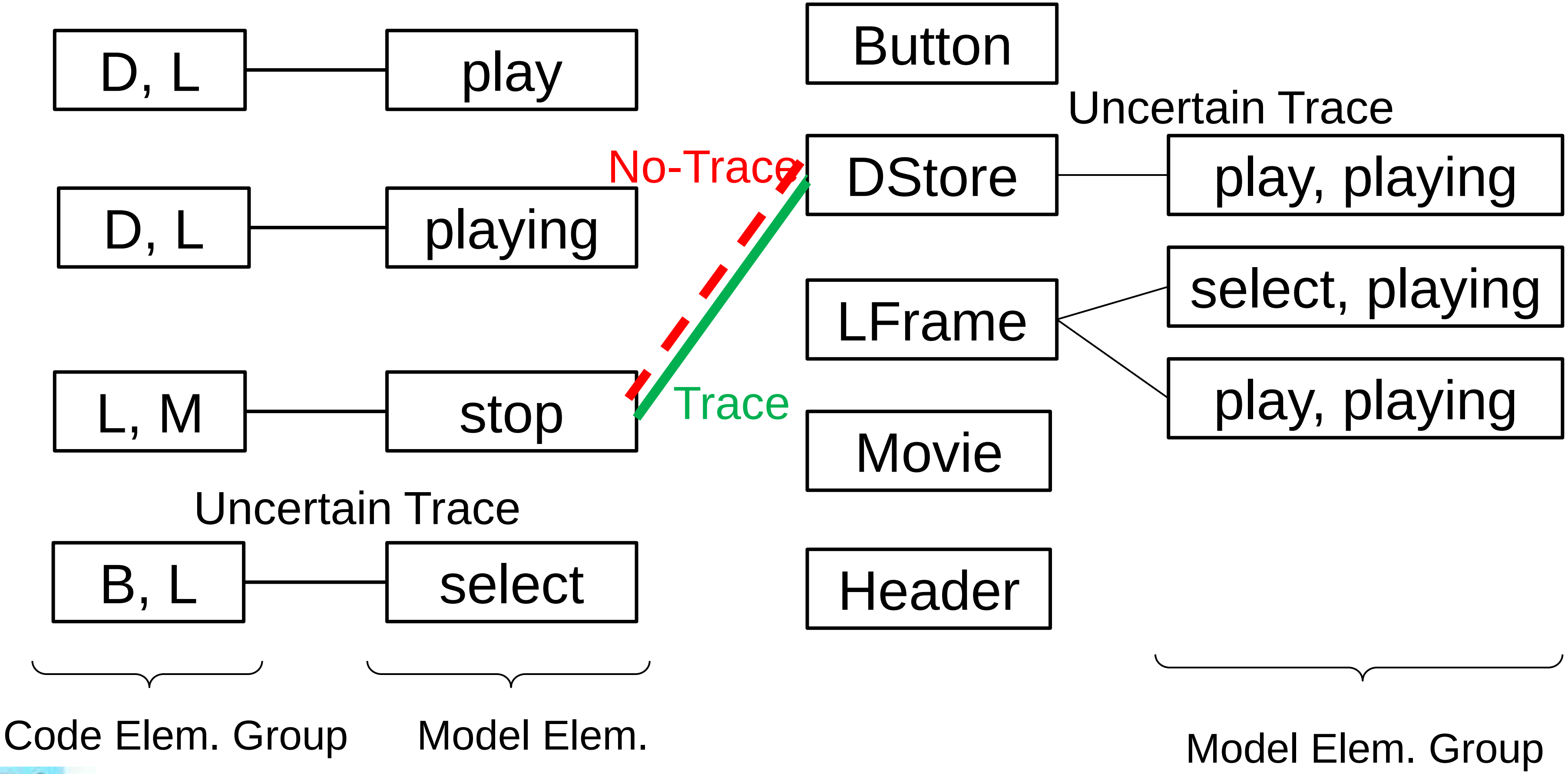
{stop} *implAtLeast* {DStore}



Correctness Constraint (1)

JOHANNES KEPLER UNIVERSITY LINZ | JKU

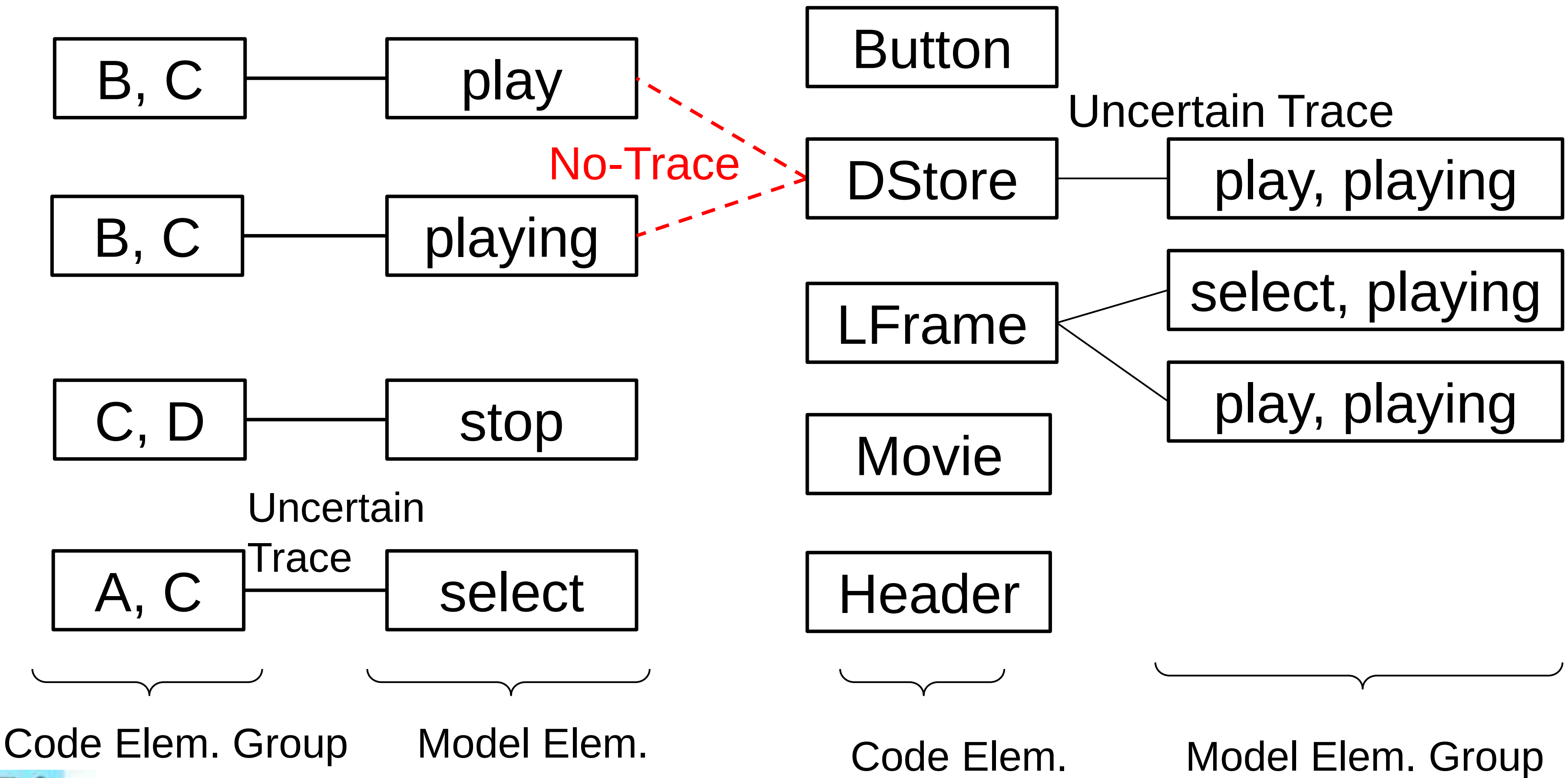
- A code element cannot be implementing and not implementing a model element at the same time



Correctness Constraints (2)

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- Every group must have at least one model element.



Validation

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- Correctness, scalability
- Evaluation of all pair wise combinations of the four types of input (implAtLeast, implAtMost, implExactly, implNot)
- 4 case study systems: ArgoUML, Siemens Route Planning, Video on demand client, and USC Inter-Library Loan

Correctness

- In most of the cases incorrect/conflicting input is detected
- The more input the more likely an incorrectness would be detected
- Incorrectness is not detected if an engineer has an incorrect but consistent understanding of the model-to-code mapping
  → unlikely when multiple engineers are working together

Scalability

- The growth of the footprint graph is polynomial with the size of the model and code
- Size of graph = #C + #M + #input * (#C + #M)
- Largest study case 30.000 nodes (ArgoUML) required less than a minute to convert the input into the footprint graph and propagate the rules for 38 ME

Summary

- Approach proposing how to describe engineers' knowledge about traceability
- Automatic detection of incorrect/inconsistent knowledge
- Automatically derive further knowledge
- Applicable to all kinds of models that are implemented in the code

Future Work

- Extend the input by adding some kind of weighting to uncertainty constructs
- Use incremental reasoning
- Apply the same technique on model-to-model traceability
- Conduct experiments on industrial projects

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Discussion

Achraf Ghabi (achraf.ghabi@jku.at)

Alexander Egyed (alexander.egyed@jku.at)

SEA *Institute for Systems Engineering and Automation*