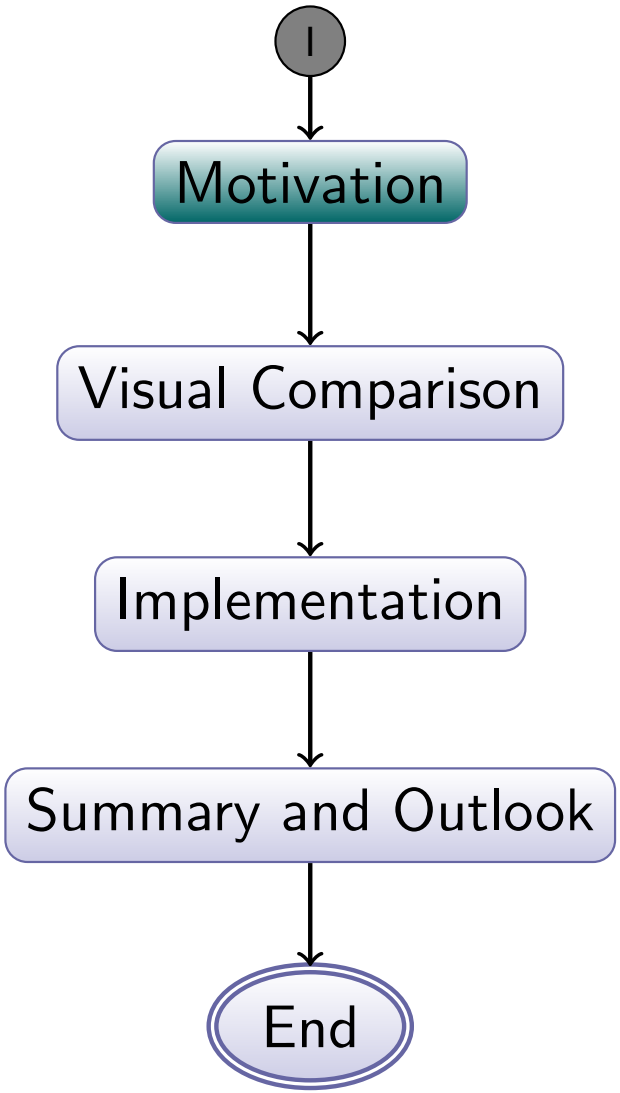
Visual Comparison of Graphical Models

Arne Schipper Hauke Fuhrmann Reinhard von Hanxleden

Real-Time Systems and Embedded Systems Group,
Department of Computer Science,
Christian-Albrechts-Universität zu Kiel
{ars,haf,rvh}@informatik.uni-kiel.de

UML&AADL'09

Outline


I
Motivation
Visual Comparison
Implementation
Summary and Outlook
End

Motivation

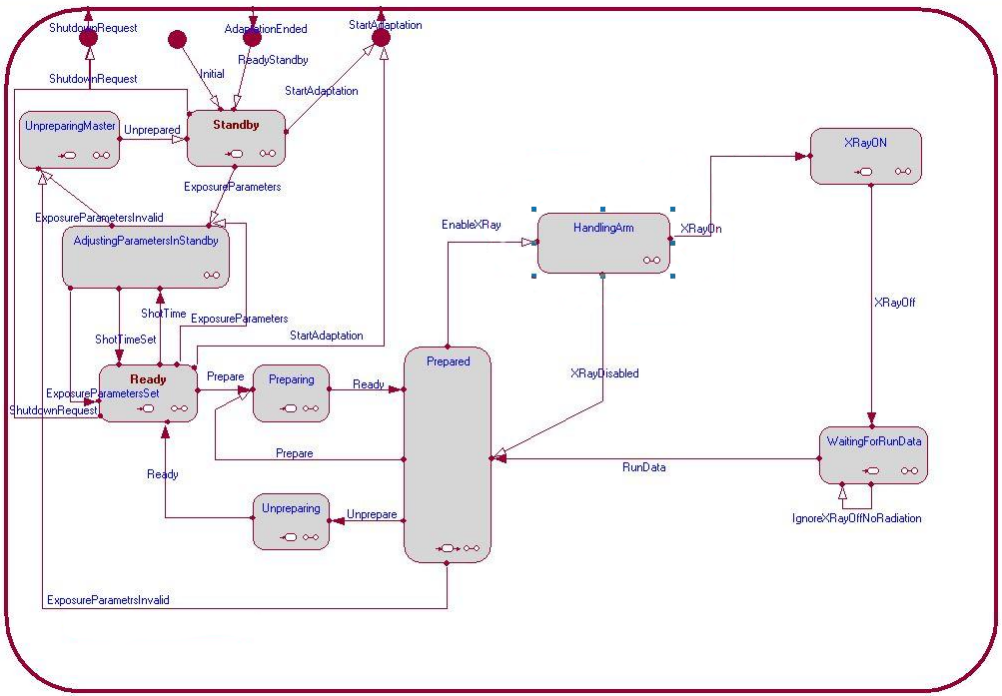
Problem:

- Graphical models very often used, quite easy to create and browse, but pain to compare.

Motivation

Problem:

- Graphical models very often used, quite easy to create and browse, but pain to compare.

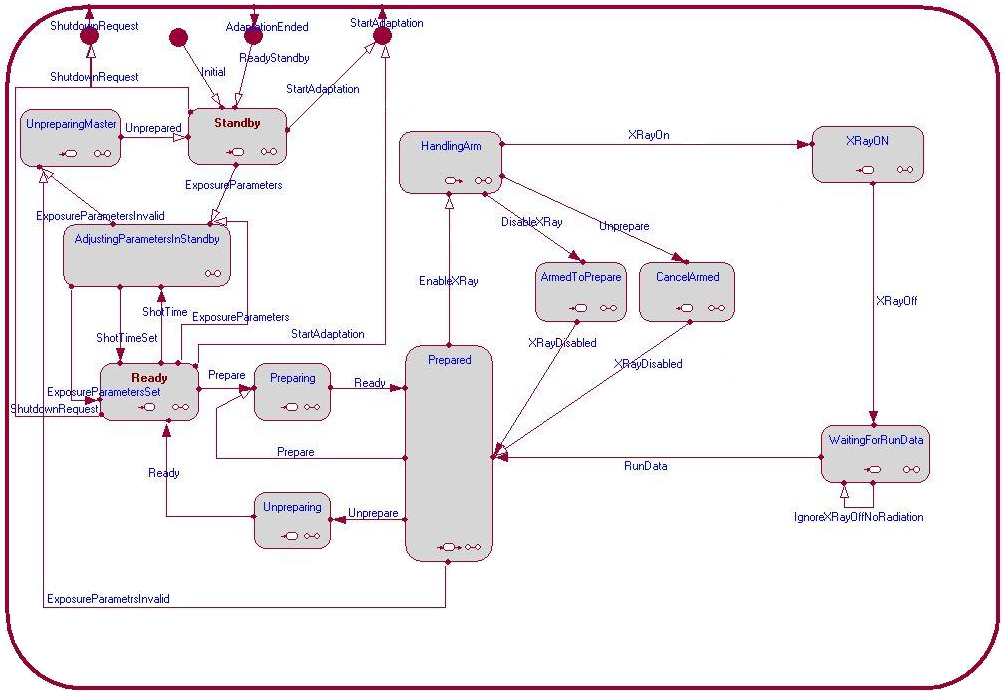


Screenshot of a real model from a project, Version 1

Motivation

Problem:

- Graphical models very often used, quite easy to create and browse, but pain to compare.

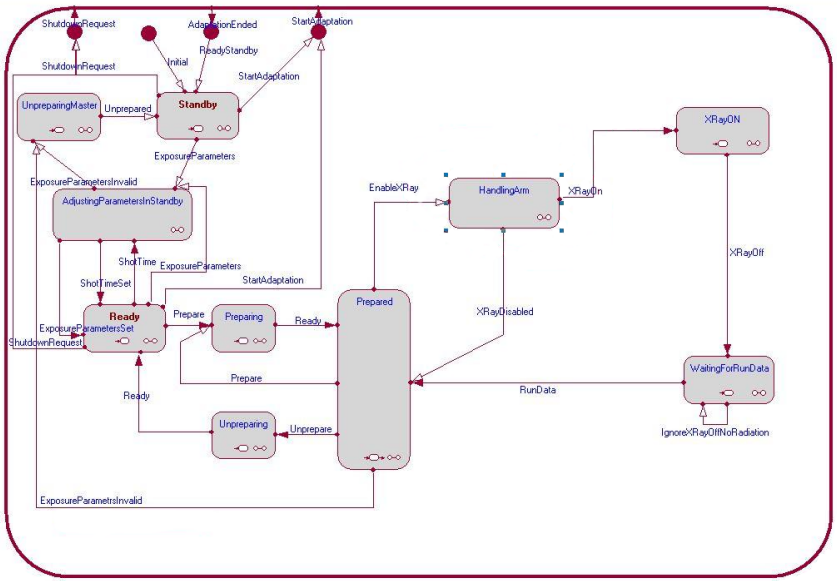


Screenshot of a real model from a project, Version 2

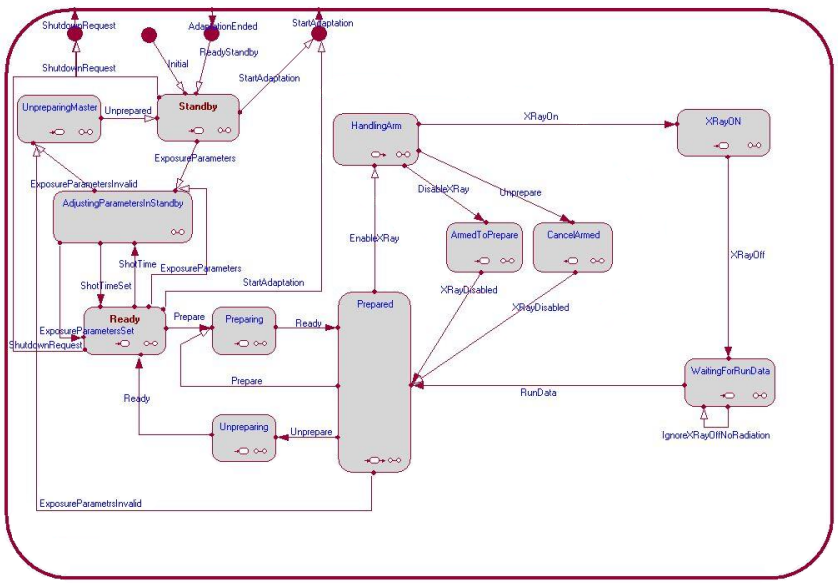
Motivation

Problem:

- Graphical models very often used, quite easy to create and browse, but pain to compare.



Screenshot of a real model from a project, Version 1



Screenshot of a real model from a project, Version 2

Motivation

Problem cont'd:

- Means exist to compare graphical models textually, but ...
- User has to switch between different abstraction levels.

Motivation

Problem cont'd:

- Means exist to compare graphical models textually, but ...
- User has to switch between different abstraction levels.

Solution:

- Develop means to aid the user in performing a real **visual comparison** of graphical models.
- Some tools exist, but have drawbacks.

Motivation

Problem cont'd:

- Means exist to compare graphical models textually, but ...
- User has to switch between different abstraction levels.

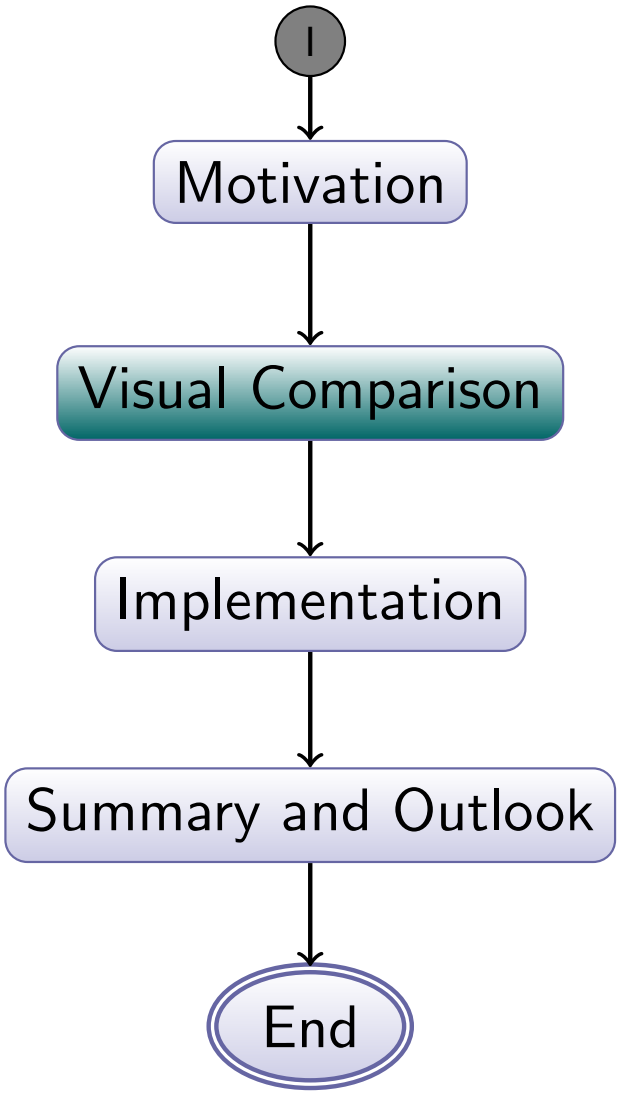
Solution:

- Develop means to aid the user in performing a real **visual comparison** of graphical models.
- Some tools exist, but have drawbacks.

Method:

- Identify and improve those drawbacks.
- Implement as Eclipse plug-in using existing techniques where appropriate.
- Use generic approach to cope with various graphical languages.

Outline


I
Motivation
Visual Comparison
Implementation
Summary and Outlook
End

Visual Comparison

Textual diff/comparison:

- Known to everyone, compare texts side by side.
- One-dimensional or linear arrangement with *holes* in the texts.

Visual Comparison

Textual diff/comparison:

- Known to everyone, compare texts side by side.
- One-dimensional or linear arrangement with *holes* in the texts.

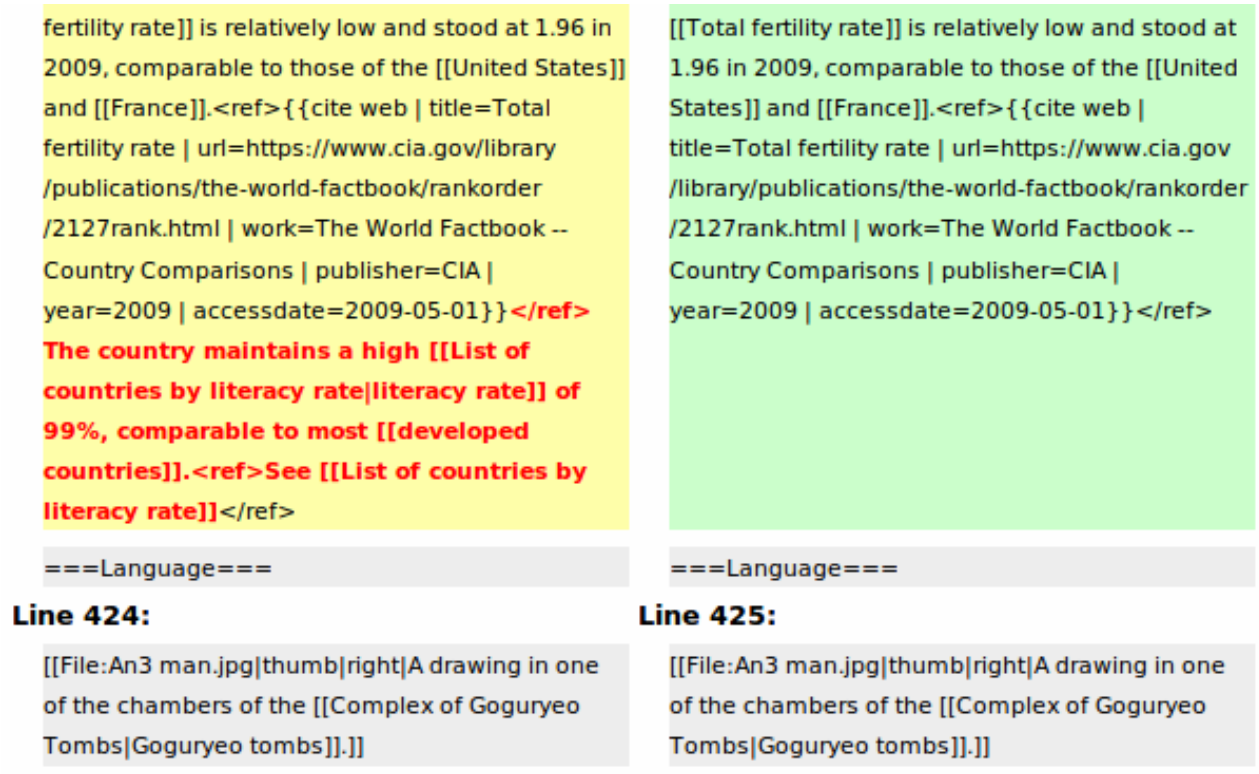


Figure: Two article versions in Wikipedia

Visual Comparison

Common comparison of graphical models:

- Generate a textual description of the changes.
- Is sometimes structured, but ...
- User has to find these changes in the graphical representation.

Visual Comparison

Common comparison of graphical models:

- Generate a textual description of the changes.
- Is sometimes structured, but ...
- User has to find these changes in the graphical representation.

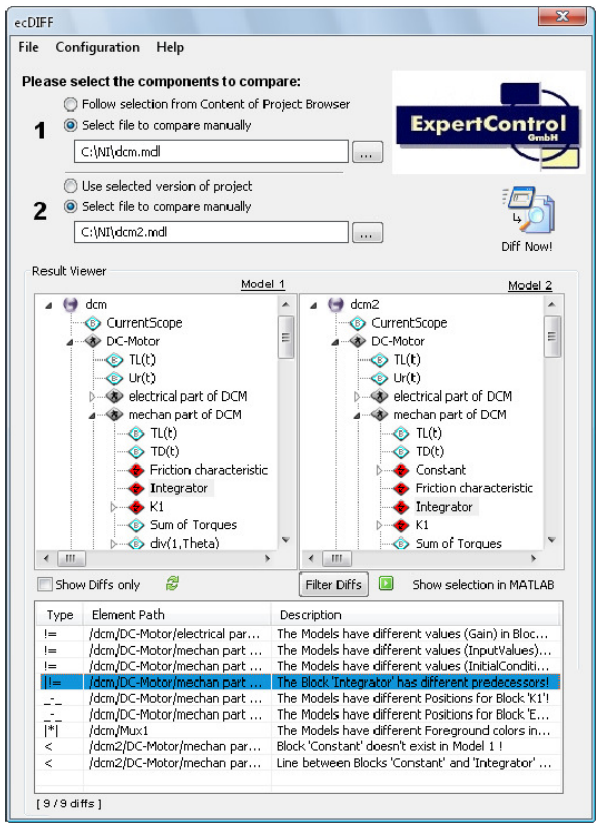


Figure: Model diff of Expert Control

Visual Comparison

Visual comparison:

- Show the changes in the graphical model itself.
- Prevents the user from switching between text and graphical model.

Visual Comparison

Visual comparison:

- Show the changes in the graphical model itself.
- Prevents the user from switching between text and graphical model.

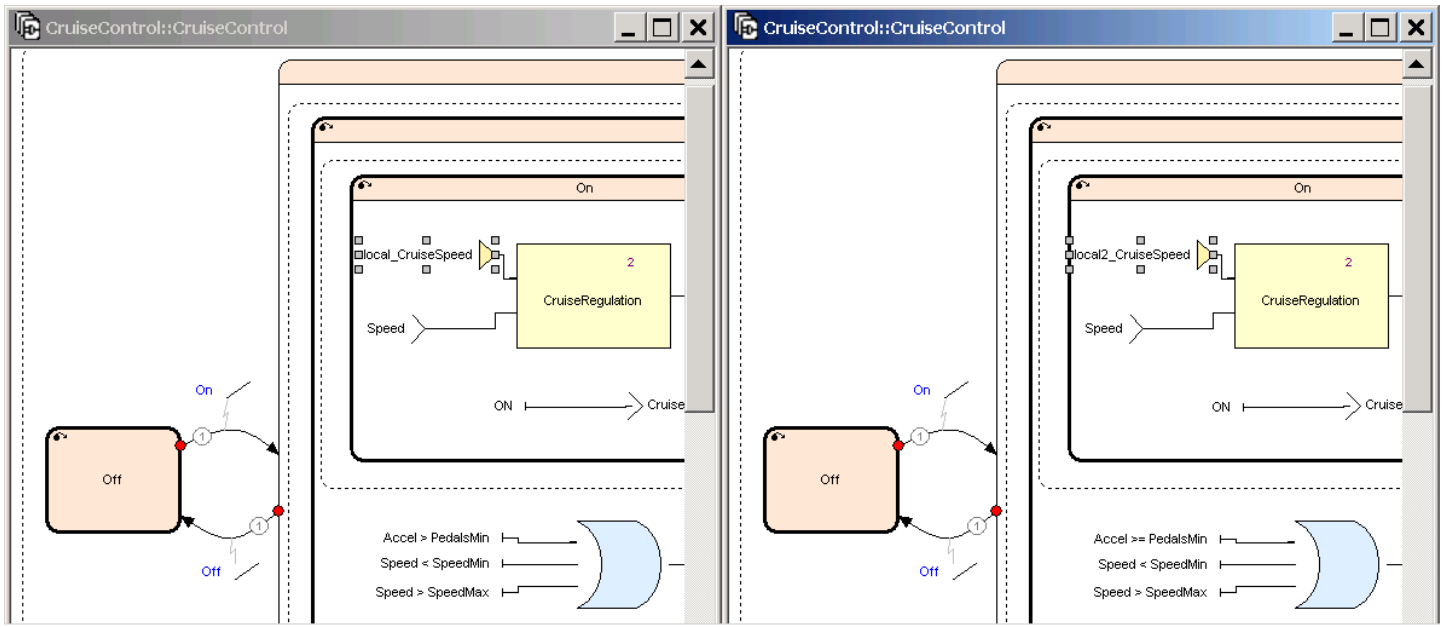


Figure: Scade model diff

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.
- Some models have information which is not shown visually.

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.
- Some models have information which is not shown visually.
- Large models.

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.
- Some models have information which is not shown visually.
- Large models.
- *Mental map* of the user.

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.
- Some models have information which is not shown visually.
- Large models.
- *Mental map* of the user.
- Difference detection. However, solved by an existing engine and we focus on graphical presentation

Questions:

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.
- Some models have information which is not shown visually.
- Large models.
- *Mental map* of the user.
- Difference detection. However, solved by an existing engine and we focus on graphical presentation

Questions:

- Use just the structure of the graphical model or also the layout information of the elements?

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.
- Some models have information which is not shown visually.
- Large models.
- *Mental map* of the user.
- Difference detection. However, solved by an existing engine and we focus on graphical presentation

Questions:

- Use just the structure of the graphical model or also the layout information of the elements?
- Use one model or both versions to display the changes?

Visual Comparison

Challenges:

- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.
- Some models have information which is not shown visually.
- Large models.
- *Mental map* of the user.
- Difference detection. However, solved by an existing engine and we focus on graphical presentation

Questions:

- Use just the structure of the graphical model or also the layout information of the elements?
- Use one model or both versions to display the changes?
- Alter the layout or leave it intact?

Visual Comparison

Challenges:

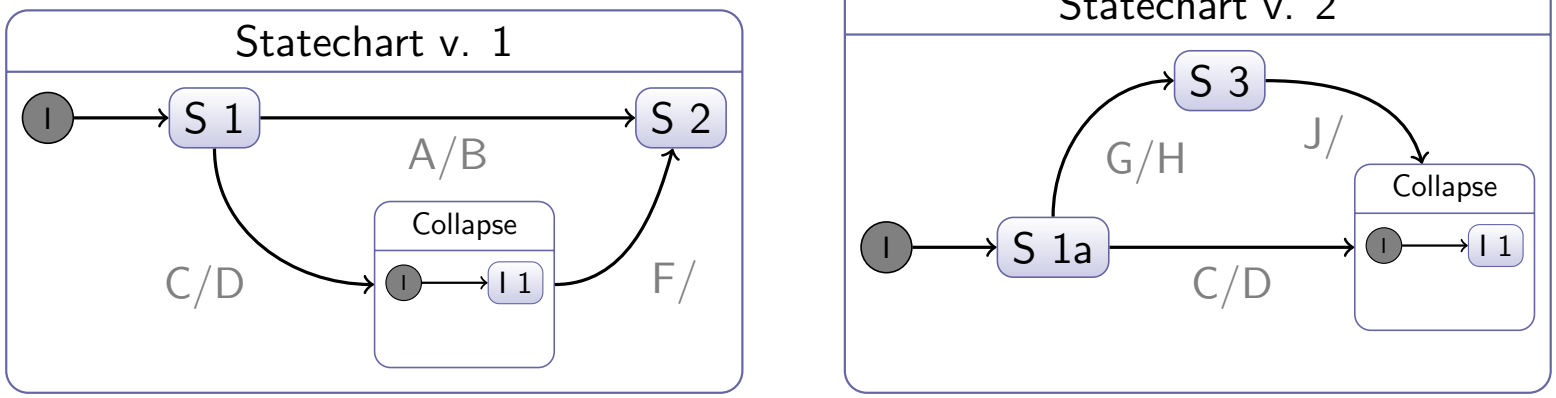
- Graphical models at least two-dimensional, in contrast to text.
- No trivial solution for *holes* like in textual diff.
- Some models have information which is not shown visually.
- Large models.
- *Mental map* of the user.
- Difference detection. However, solved by an existing engine and we focus on graphical presentation

Questions:

- Use just the structure of the graphical model or also the layout information of the elements?
- Use one model or both versions to display the changes?
- Alter the layout or leave it intact?
- Does a readable automatic layout help?

Visual Comparison

The two versions of the model:



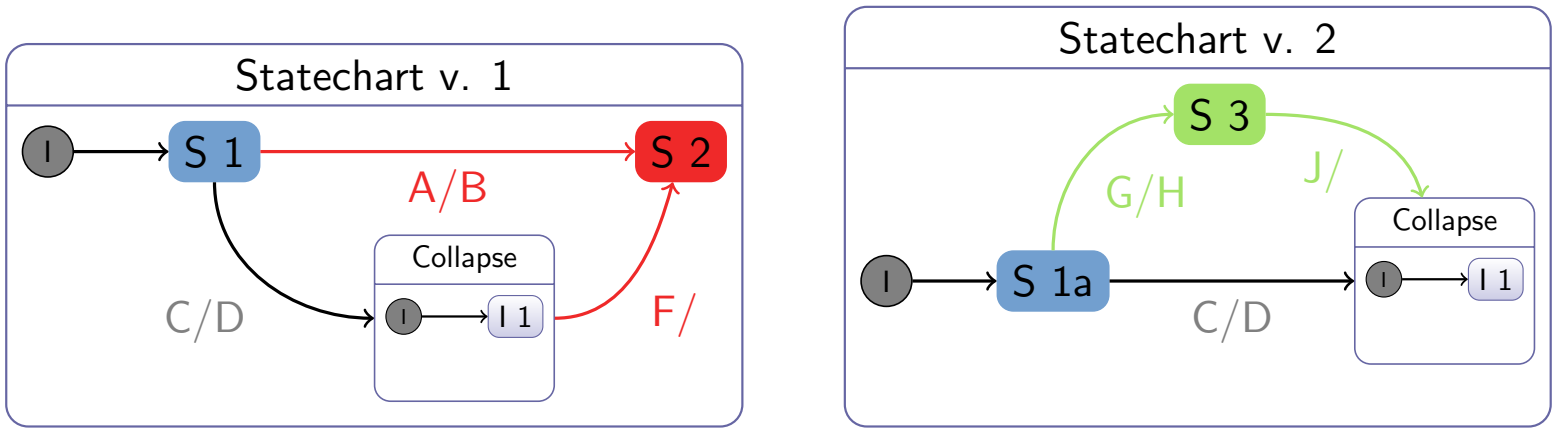
(a) Version 1

(b) Version 2

Figure: The two original versions of the example diagram.

Visual Comparison

Possible representation of the changes 1:



(a) Version 1

(b) Version 2

Figure: Plain visual diff. Color legend: green/additions, red/deletions, blue/changes.

Visual Comparison

Possible representation of the changes 3:

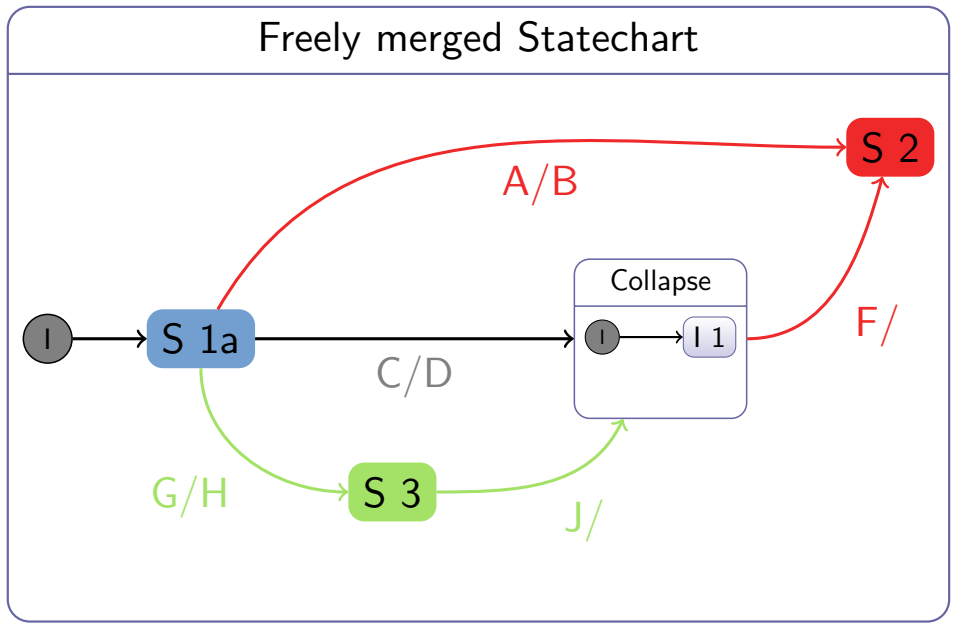


Figure: Freely merged visual diff.

Visual Comparison

Which representation?

- Manual tests showed that *plain* visual diff is best.

Visual Comparison

Which representation?

- Manual tests showed that *plain* visual diff is best.
- Additional textual description of changes is also given.

Visual Comparison

Which representation?

- Manual tests showed that *plain* visual diff is best.
- Additional textual description of changes is also given.
- No problems with/recomputation of layout, but a good layout of the original models is helpful.

Visual Comparison

Which representation?

- Manual tests showed that *plain* visual diff is best.
- Additional textual description of changes is also given.
- No problems with/recomputation of layout, but a good layout of the original models is helpful.
- Mental map of user is preserved.

Visual Comparison

Which representation?

- Manual tests showed that *plain* visual diff is best.
- Additional textual description of changes is also given.
- No problems with/recomputation of layout, but a good layout of the original models is helpful.
- Mental map of user is preserved.
- Additional means like panning, zooming and folding needed to cope with large models.

Visual Comparison

Which representation?

- Manual tests showed that *plain* visual diff is best.
- Additional textual description of changes is also given.
- No problems with/recomputation of layout, but a good layout of the original models is helpful.
- Mental map of user is preserved.
- Additional means like panning, zooming and folding needed to cope with large models.

Other issues:

- The diff is performed just against the structural/domain model.

Visual Comparison

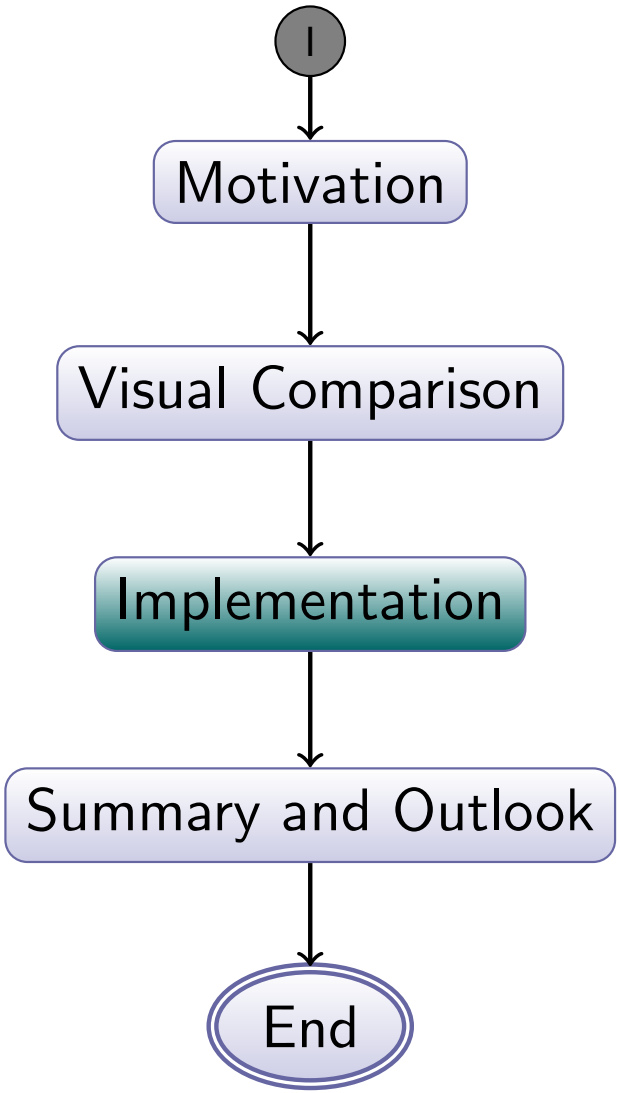
Which representation?

- Manual tests showed that *plain* visual diff is best.
- Additional textual description of changes is also given.
- No problems with/recomputation of layout, but a good layout of the original models is helpful.
- Mental map of user is preserved.
- Additional means like panning, zooming and folding needed to cope with large models.

Other issues:

- The diff is performed just against the structural/domain model.
- Non graphical changes (e.g. of properties) are also displayed; blue in the previous slides.

Outline


I
Motivation
Visual Comparison
Implementation
Summary and Outlook
End

Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.

Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.
- GMF to build the corresponding graphical editor.

Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.
- GMF to build the corresponding graphical editor.
- EMF Compare to compute the differences of the EMF model.

Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.
- GMF to build the corresponding graphical editor.
- EMF Compare to compute the differences of the EMF model.
- **KiViK** (Kieler Visual Comparison) to get EMF Compare output into GMF.

Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.
- GMF to build the corresponding graphical editor.
- EMF Compare to compute the differences of the EMF model.
- **KiViK** (Kieler Visual Comparison) to get EMF Compare output into GMF.
  - Use original layout of diagrams and display them side by side.

Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.
- GMF to build the corresponding graphical editor.
- EMF Compare to compute the differences of the EMF model.
- **KiViK** (Kieler Visual Comparison) to get EMF Compare output into GMF.
  - Use original layout of diagrams and display them side by side.
  - Annotate the structural changes with different colors.

Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.
- GMF to build the corresponding graphical editor.
- EMF Compare to compute the differences of the EMF model.
- **KiViK** (Kieler Visual Comparison) to get EMF Compare output into GMF.
  - Use original layout of diagrams and display them side by side.
  - Annotate the structural changes with different colors.
  - Use third panel on top to display just the structural changes textually (like EMF Compare).

Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.
- GMF to build the corresponding graphical editor.
- EMF Compare to compute the differences of the EMF model.
- **KiViK** (Kieler Visual Comparison) to get EMF Compare output into GMF.
  - Use original layout of diagrams and display them side by side.
  - Annotate the structural changes with different colors.
  - Use third panel on top to display just the structural changes textually (like EMF Compare).
  - Equip the comparison view with means to navigate and zoom.

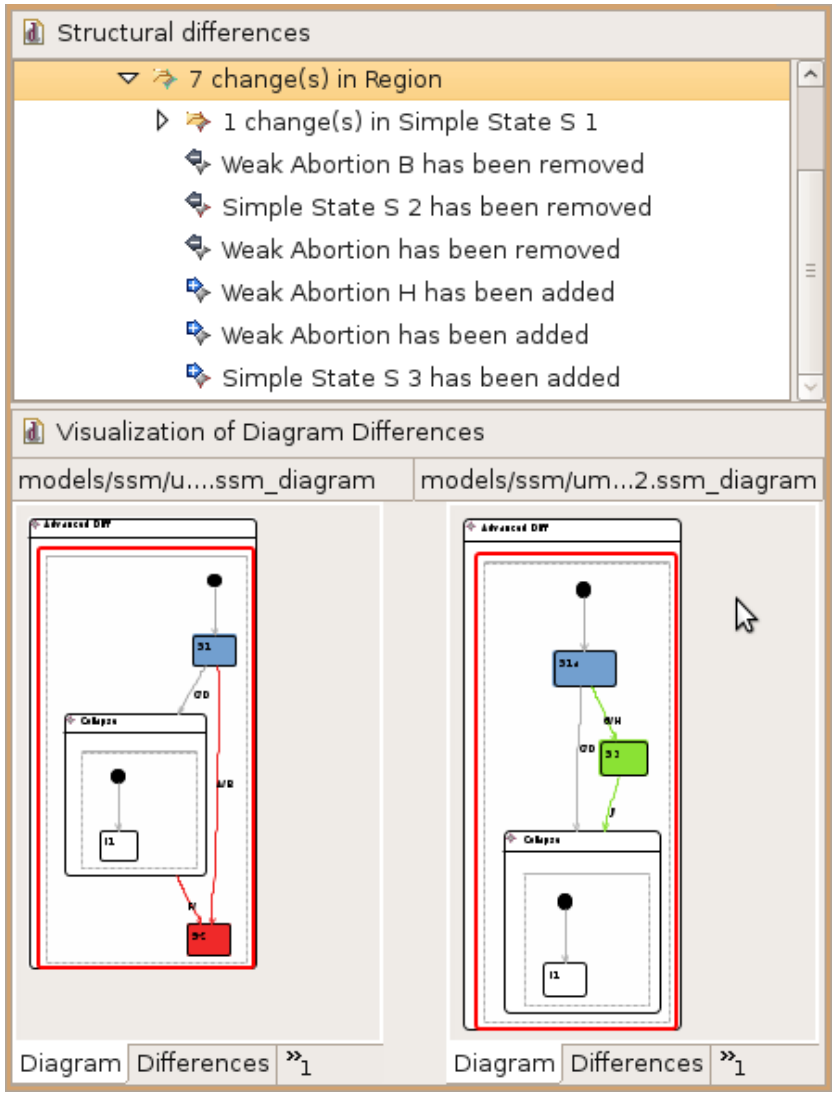
Implementation

Developed as Eclipse plug-in within a project called KIELER (Kiel Integrated Environment for Layout, for Eclipse RCP).

- EMF to create the domain models.
- GMF to build the corresponding graphical editor.
- EMF Compare to compute the differences of the EMF model.
- **KiViK** (Kieler Visual Comparison) to get EMF Compare output into GMF.
  - Use original layout of diagrams and display them side by side.
  - Annotate the structural changes with different colors.
  - Use third panel on top to display just the structural changes textually (like EMF Compare).
  - Equip the comparison view with means to navigate and zoom.
  - Collapse composite items with no changes inside (a layout algorithm is needed then) .

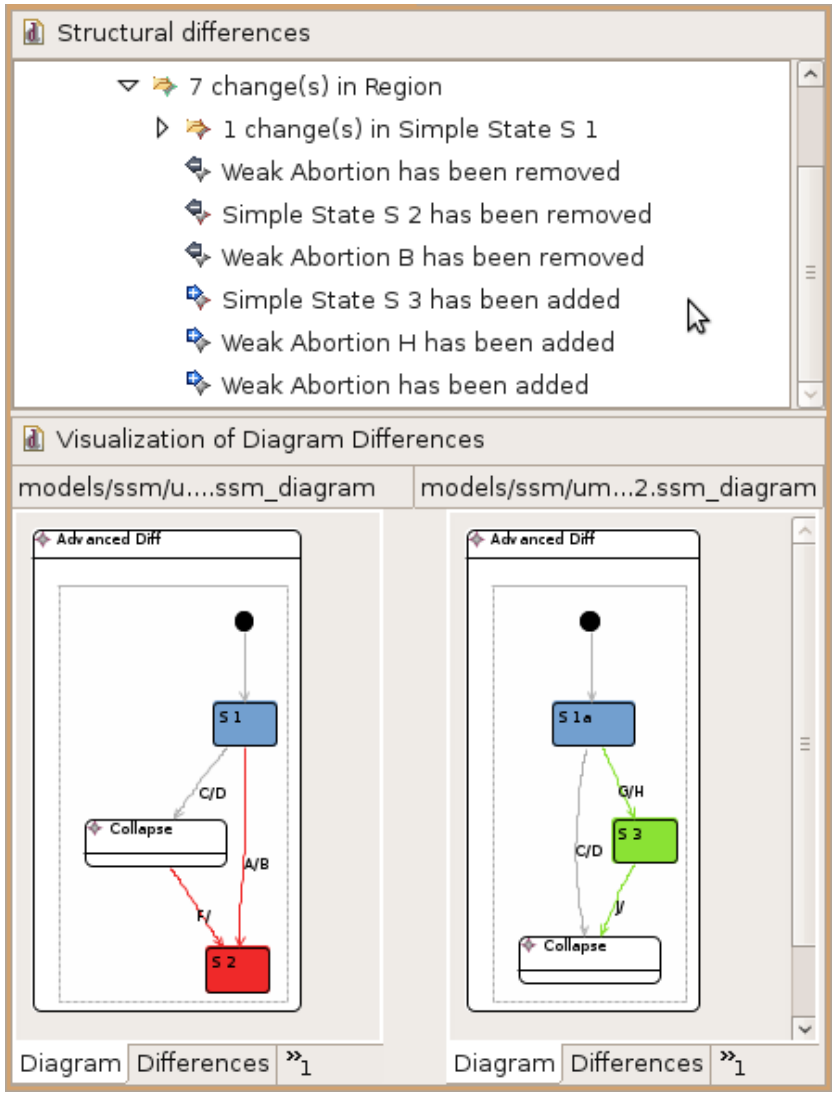
Implementation

General implementation:



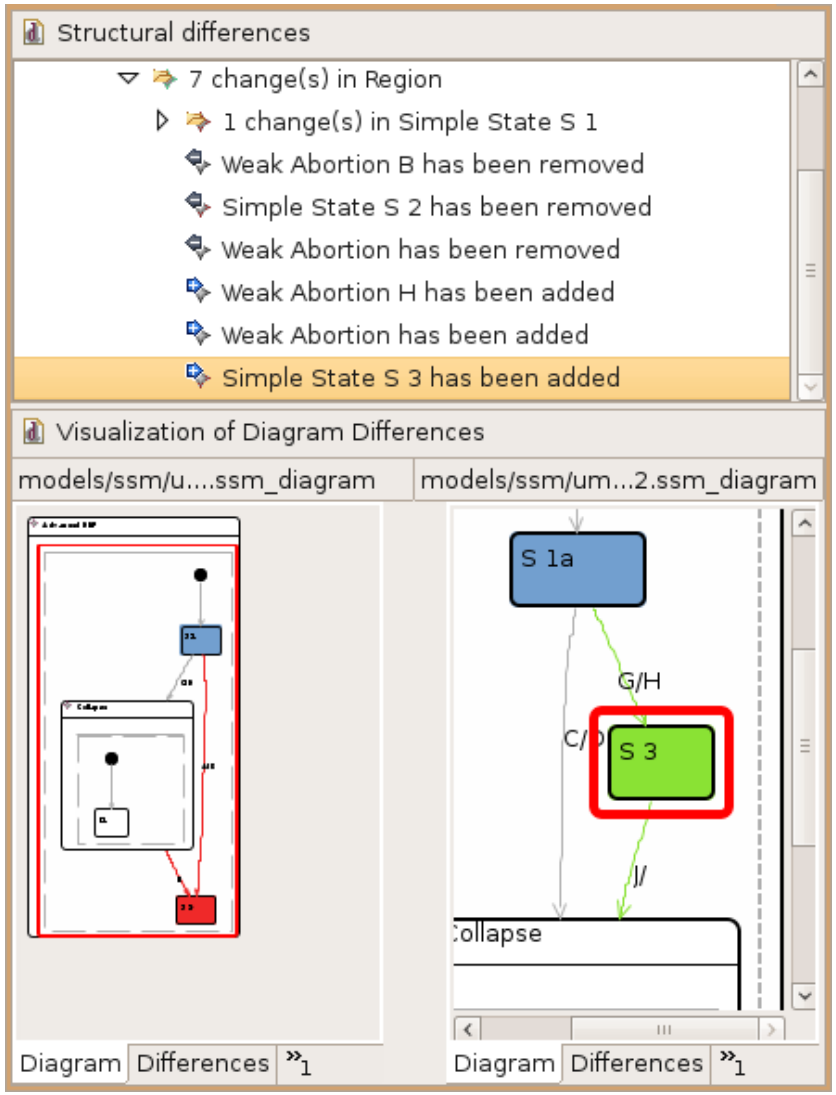
Implementation

Example of collapsing:



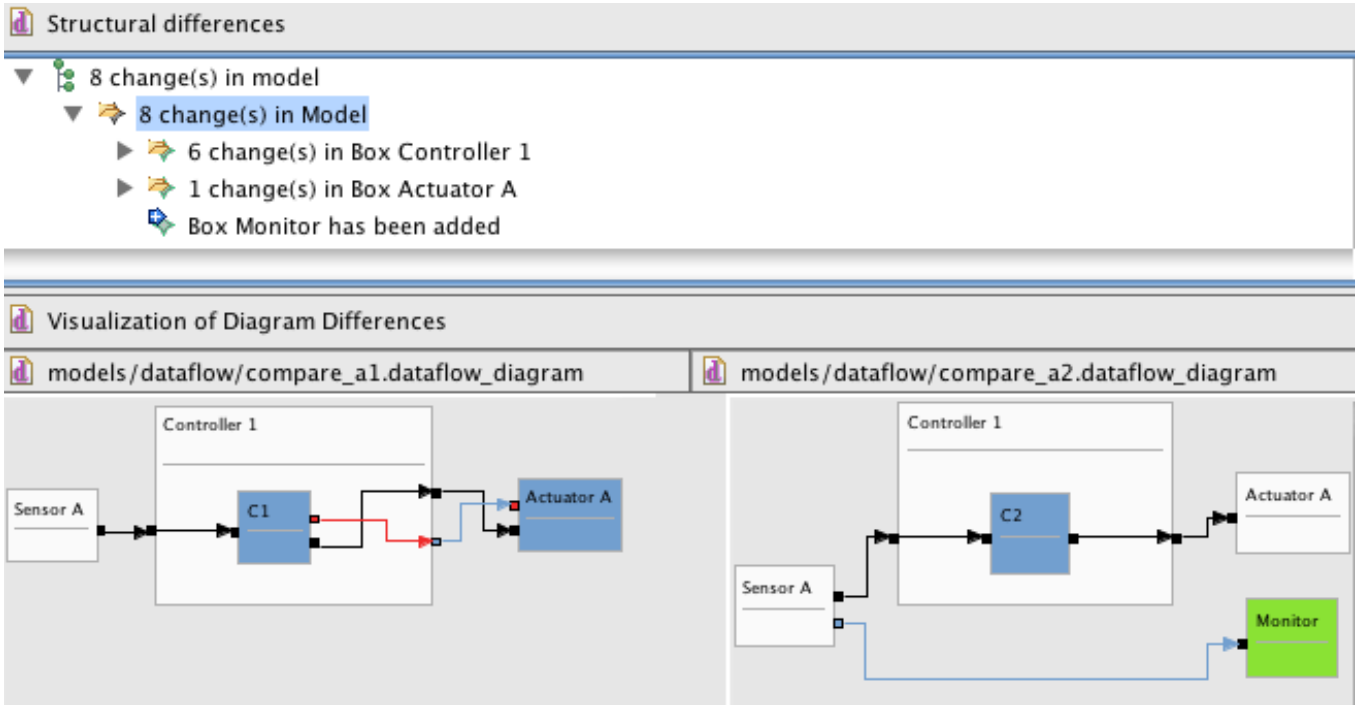
Implementation

Example of automatic zoom:

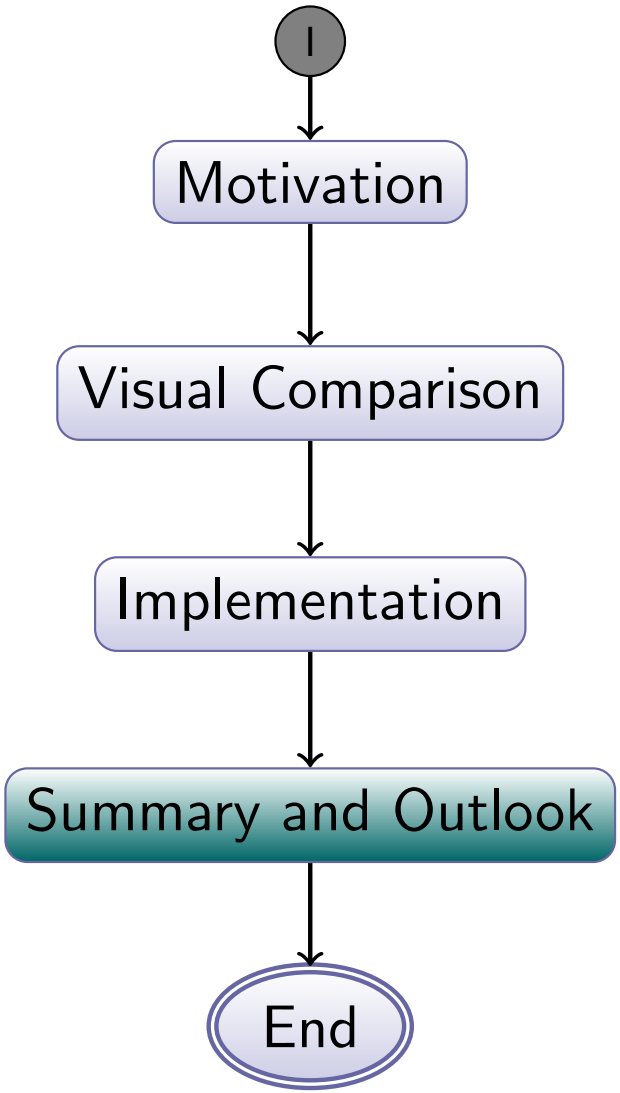


Implementation

Comparison of Dataflow models:



Outline


I
Motivation
Visual Comparison
Implementation
Summary and Outlook
End

Summary and Outlook

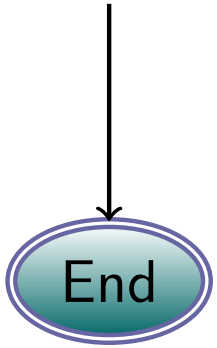
Feedback:

- Students and professionals gave an overall positive feeback for this approach.
- Representation directly in the diagram seen as benefit.
- Visualization of small (or invisible) changes very useful.
- User interface with collapsing, panning and zooming intuitive.
- Generic approach enables support for various diagrams with none or little adaption.

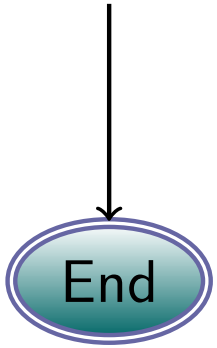
Summary and Outlook

Outlook:

- Large models are still challenging; time for comparison as well as navigation.
- Next step would be to support merging graphically.
- Maybe implement also the other approaches presented to see how they perform.



Visual Comparison of Graphical Models



Visual Comparison of Graphical Models

Thanks!