Evolution of Model Clones in Simulink

Models and Evolution 2013

Matthew Stephan & Manar H. Alalfi & James R. Cordy & Andrew Stevenson
Queen's University, Ontario, Canada

Why was this work started?

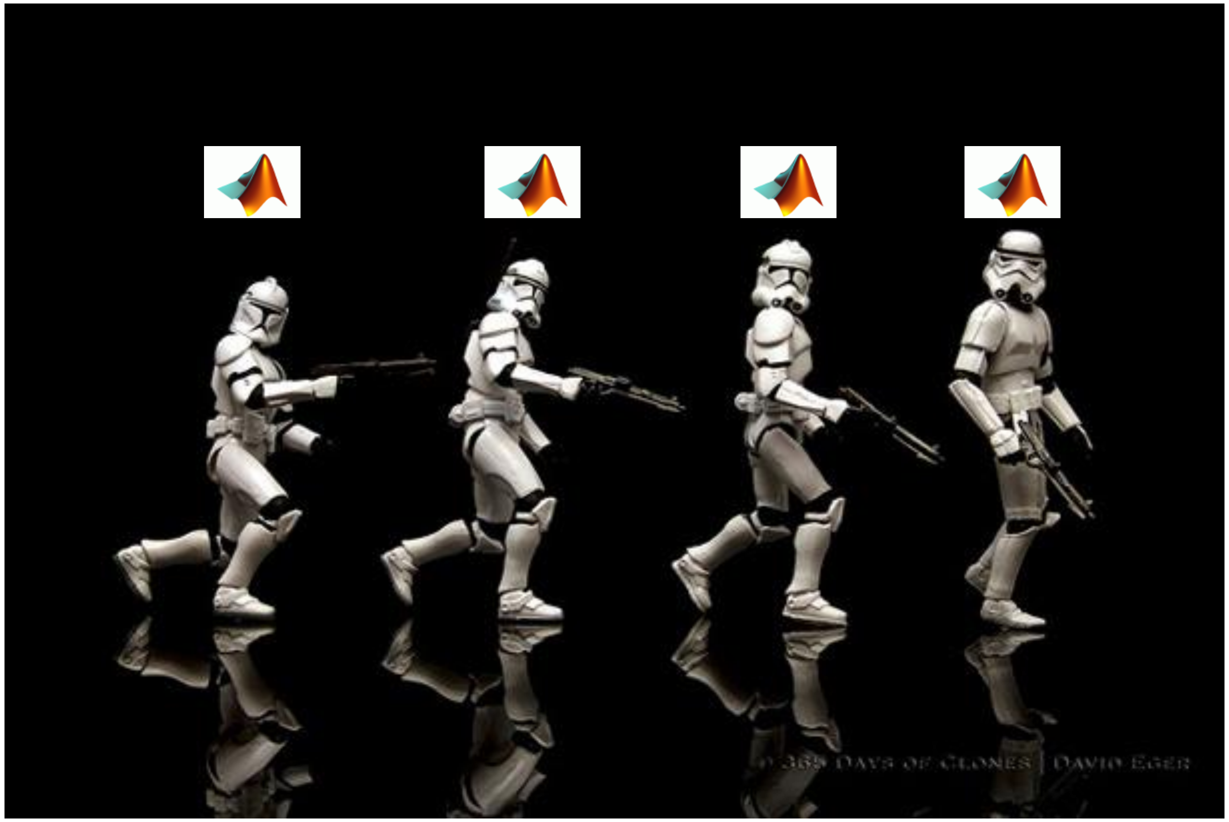
Model evolution for UML is relatively mature.

In contrast, model evolution for Simulink and other data-flow languages -> ~ Non Existent

Simulink is very different from UML.
Closest match = UML activity diagram.

In the past, code clone evolution has been used to assist in understanding source code evolution.

Simulink Clone Evolution



Overview of what we do

Detect Clones

- Run SIMONE on 3 systems (2 open source, 1 public)

Develop SIMCCT

- Simulink Clone Class Tracker (SIMCCT)
- Able to track evolution of a model-clone class' clone instances throughout multiple versions

Take note of MCC evolution

- Observe MCC's clone instances in future version
- Note the relationships between clone instances in original set

Investigate model evolution causing MCC changes

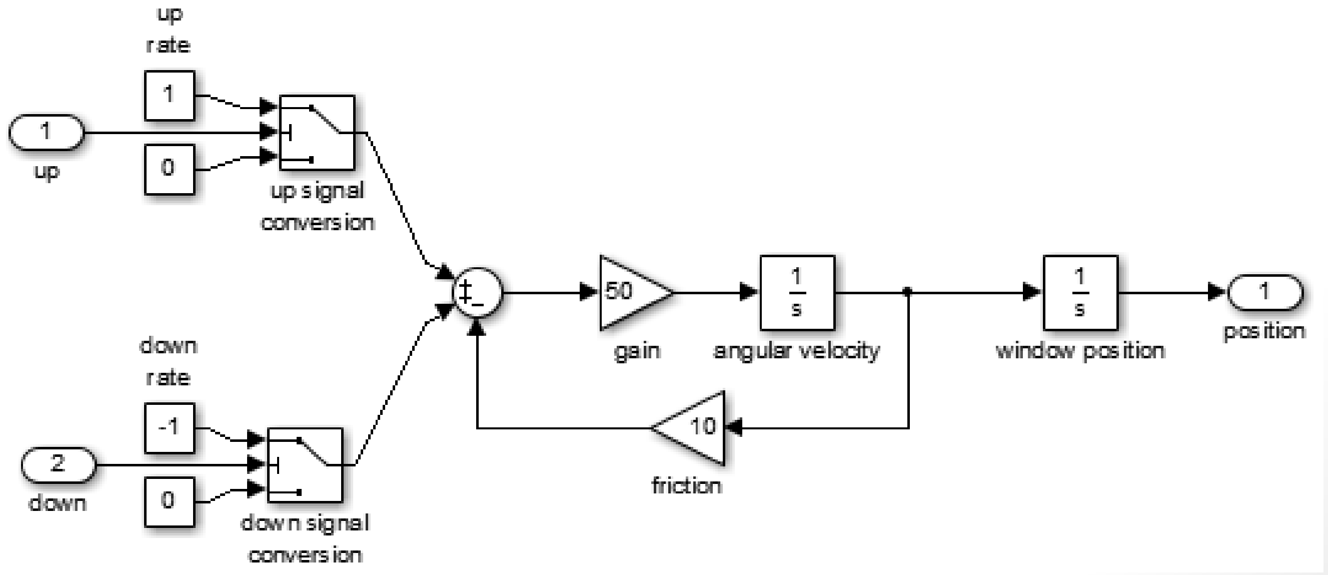
- When there is a change from 1 version to the next, look at what model evolution has transpired to cause it.

Background – Simulink

Consists of 3 granularity levels

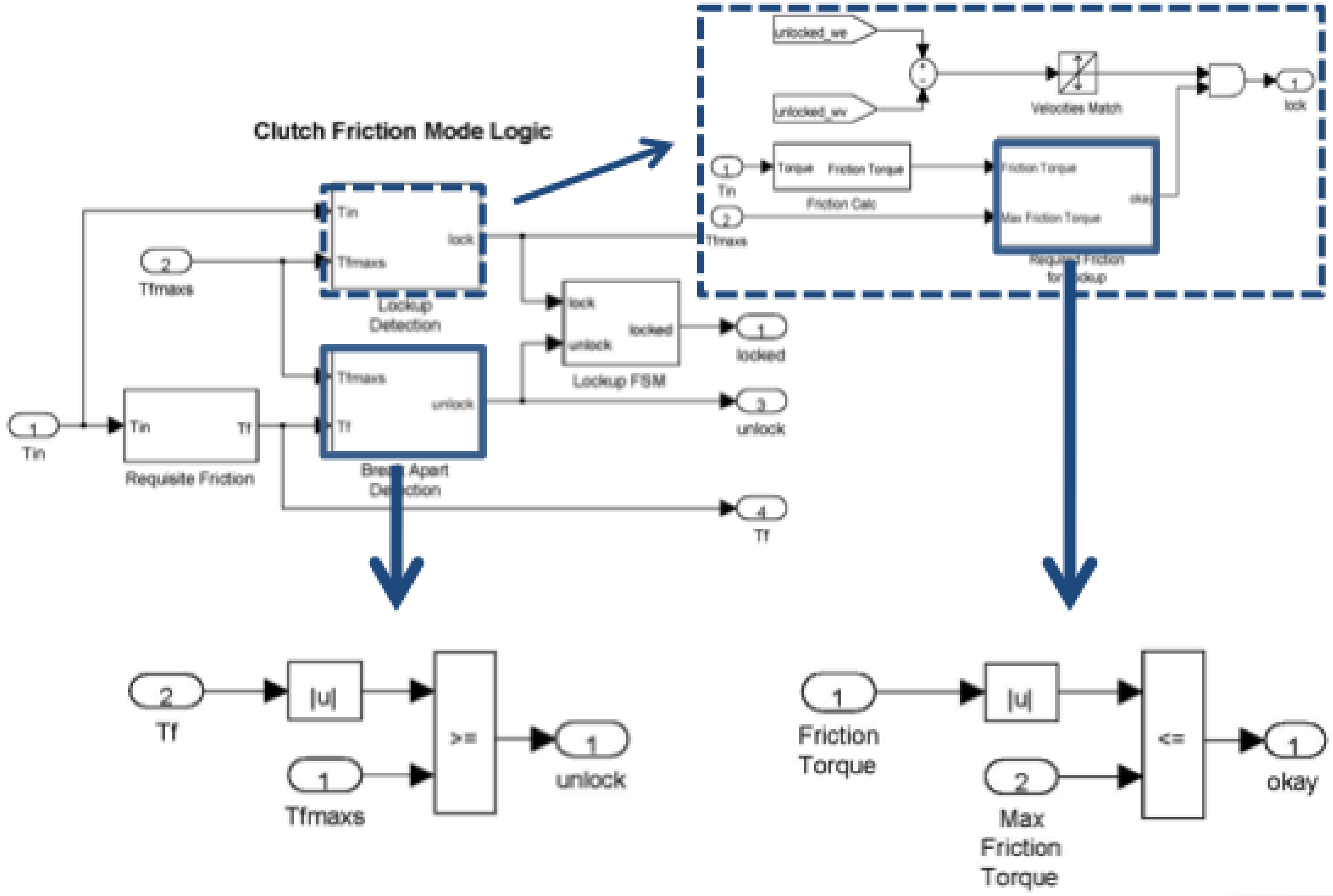
- Models
- (Sub) systems
- Blocks

All block names in a model must be unique and contain >= 1 character



Background – Model Clone Detection and Simone

- MCD = Finding identical or similar fragments of model elements
- Simone = Text-based MCD approach



Background – Clone Genealogies

Genealogy for code clone groups = Way in which a collection of clones evolve

- Clone group evolution they describe is in terms of code snippets = text and a location.

In exact code clones, matching groups to other clone groups based on textual similarity.

In near-miss code clones, match groups to other groups using function containment.

Definitions

MCI

- Model Clone Instance
- Contains list of blocks, the list of lines, and its location
- Location = model and system(s) the MCI is contained in.

MCC

- Model Clone Class
- A collection of MCIs that are grouped together by a model clone detector based on some measure of classification

Tracing MCIs Across Versions

- Can not trace classes for near-miss clones, as proved by Saha et al., so we trace MCIs.
- To trace MCIs, we use
  1. The model containing the MCI
  2. Fully qualified path to the system(s) comprising the MCI.
     - Because all blocks, including systems, which are blocks of type ``subsystem'', must have unique names -> suitable source of traceability.
     - Analogous to Saha et al.'s code clone group mapping in which they determine if a code clone fragment is contained within a function.

SIMCCT

Allows a user to select an MCC from any version

- Shows, in a GUI, what MCCs in future versions contain its MCIs.

SIMCCT takes in a set of XML MCD results in the same order as the versions they correspond to.

- For the input thus for, wrote a TXL transformation in order to change the output from Simone into a form more conducive for evolution analysis.

Input format

```
<clones>
    <class classid="#" nclones="#" similarity="#" ...>
       <source file="..."subsystem="..." ... >
           <block path="..." type="..." ...Block attributes.../>
           ...More Blocks...
           <line ...Line attributes"/>
              ...More Lines...
       </source>
                ...More Sources...
     </class>
     ...More Classes...
</clones>
```

SIMCCT Steps

Parse and Extract

- Parses the file, extracting what it needs.
- Treats each file as a version

Identify unique MCIs

- Identifies unique MCIs across all versions assigns them a unique ID number.
- ID used in GUI since text = Too long

Relate MCCs to Future MCCs

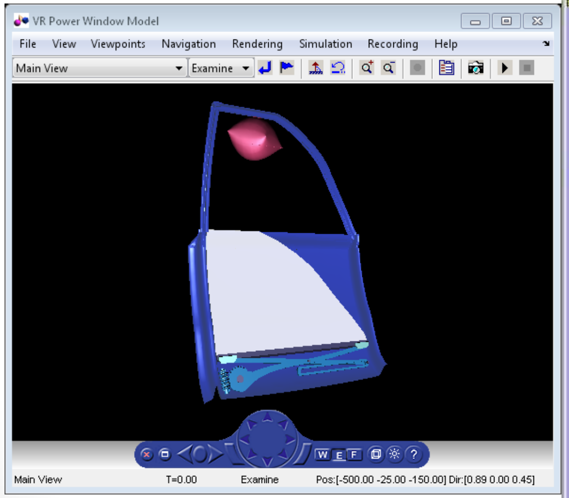
- Each time MCC is selected by user, relate and display future MCCs.

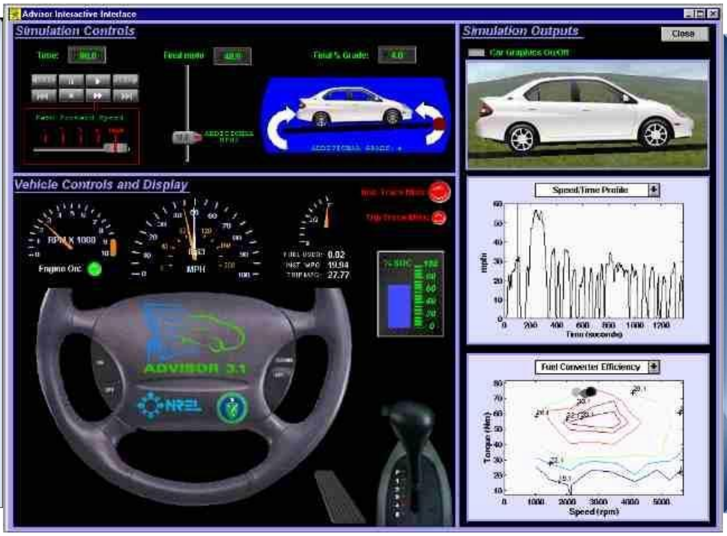
SIMCCT – Relating MCCs

- Search for the MCCs in successive versions that contain the MCIs belonging to the selected MCC
    1. Consider an MCC with class ID 4 from a first version, MCC{v1c4}.
    2. It is selected and contains a set of MCIs, MCI{v1c4}.
    3. In future version 'x' and potential clone class 'y', MCC{vxcy} is displayed if MCI{vxcy} contains any element from MCI{v1c4}.
    - Examples with pictures coming will demonstrate
- Indicate relationship to future MCCs and use as starting point to (manually) look into model evolution that caused MCC shift.

Experiment – Systems under Study

- Ran SIMCCT on 3 systems
  - Power Window from Simulink Demo set
  - Advanced Vehicle Simulator
    - large open source system
  - GM Models


VR Power Window Model
File View Viewpoints Navigation Rendering Simulation Recording Help
Main View
Examine
Main View T=0.00 Examine Pos:[-500.00 -25.00 -150.00] Dir:[0.89 0.00 0.45]



Experiment – Systems under Study

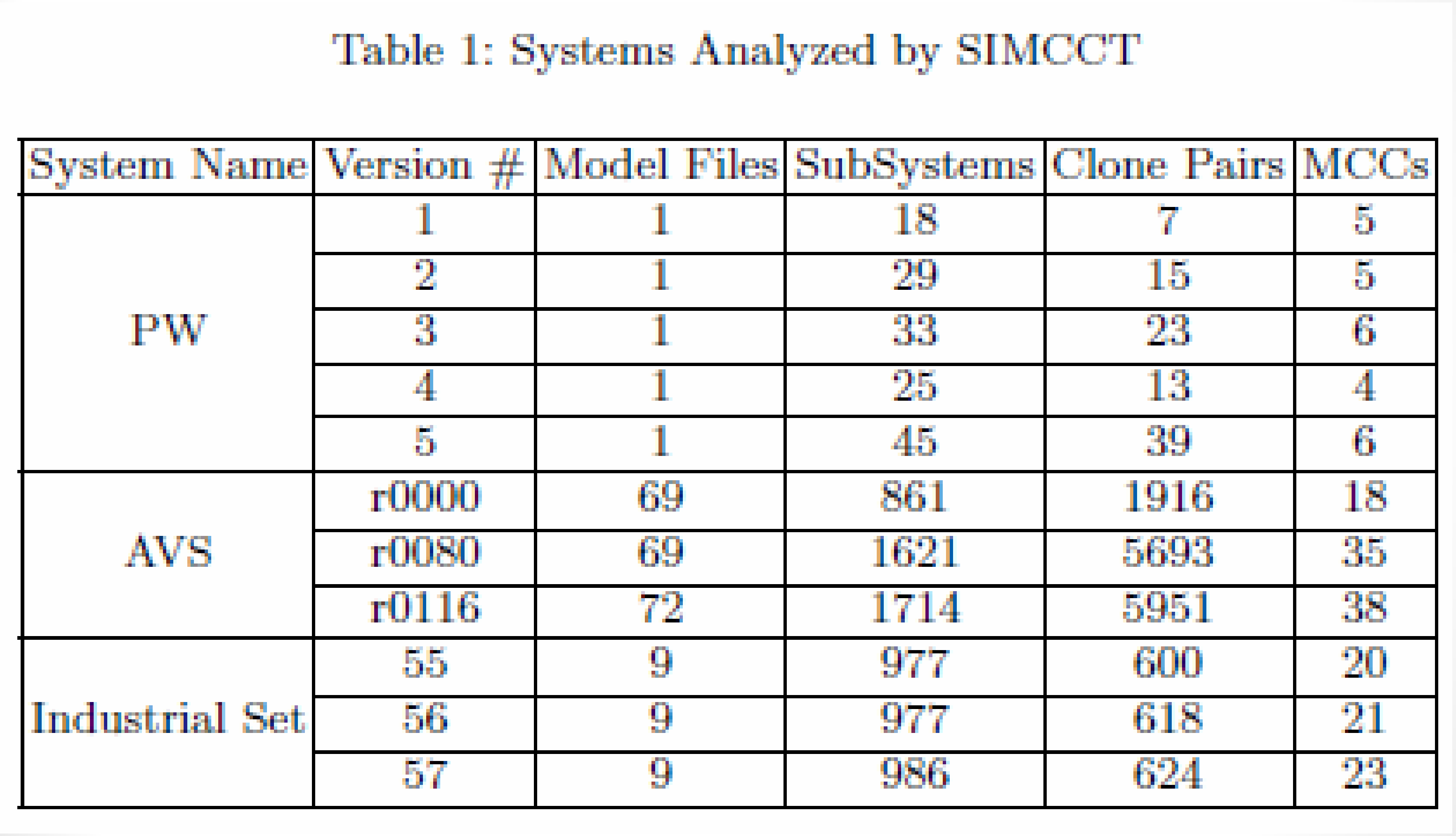


Table 1: Systems Analyzed by SIMCCT

System Name	Version #	Model Files	SubSystems	Clone Pairs	MCCs
PW	1	1	18	7	5
	2	1	29	15	5
	3	1	33	23	6
	4	1	25	13	4
	5	1	45	39	6
AVS	r0000	69	861	1916	18
	r0080	69	1621	5693	35
	r0116	72	1714	5951	38
Industrial Set	55	9	977	600	20
	56	9	977	618	21
	57	9	986	624	23

SIMCCT Results at a Glance

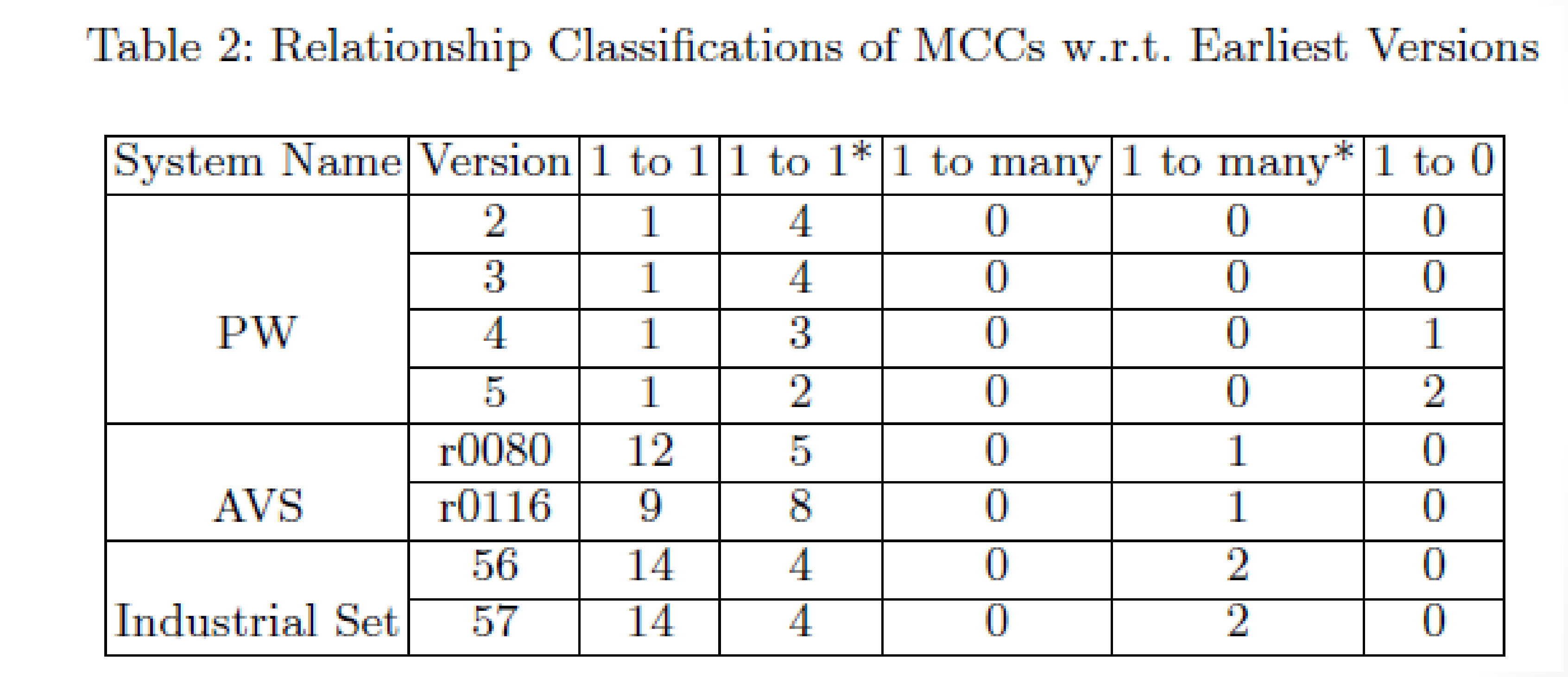


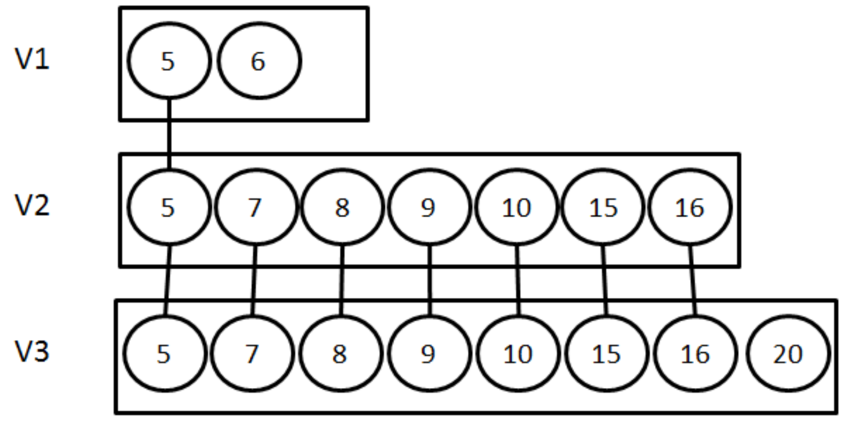
Table 2: Relationship Classifications of MCCs w.r.t. Earliest Versions

System Name	Version	1 to 1	1 to 1*	1 to many	1 to many*	1 to 0
PW	2	1	4	0	0	0
	3	1	4	0	0	0
	4	1	3	0	0	1
	5	1	2	0	0	2
AVS	r0080	12	5	0	1	0
	r0116	9	8	0	1	0
Industrial Set	56	14	4	0	2	0
	57	14	4	0	2	0

Examples

- Use similar representation to Gode for type-1 clones.
  - MCCs = rectangles
  - MCIs = circles
- Choose examples from public models as they adequately exhibit the cases and are available to all.
- Also investigate what evolution has taken place on the models themselves that caused the observed MCE.
- Each number within a circle refers to a uniquely identified key that corresponds to a unique MCI across all versions.
  - (Kept this information for reproducibility and for referring to in text)

Power Window – MCC 3

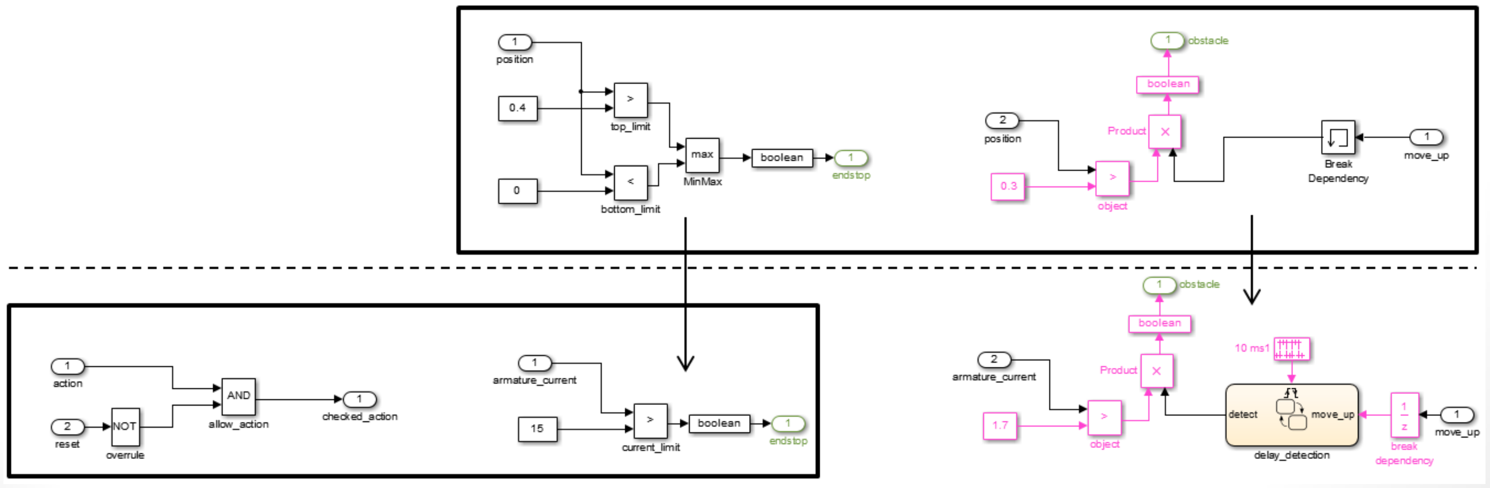


V4 & V5 Does not belong to any MCC

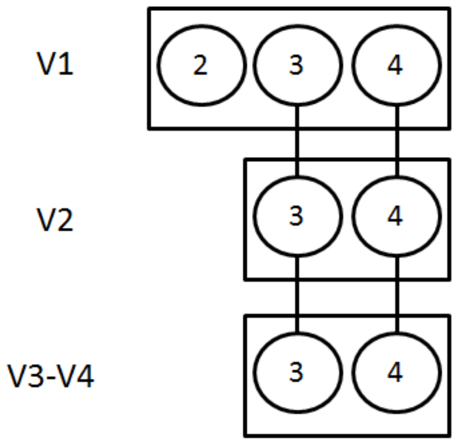
Check_UP
ID 7

Detect_Endstop
ID 5

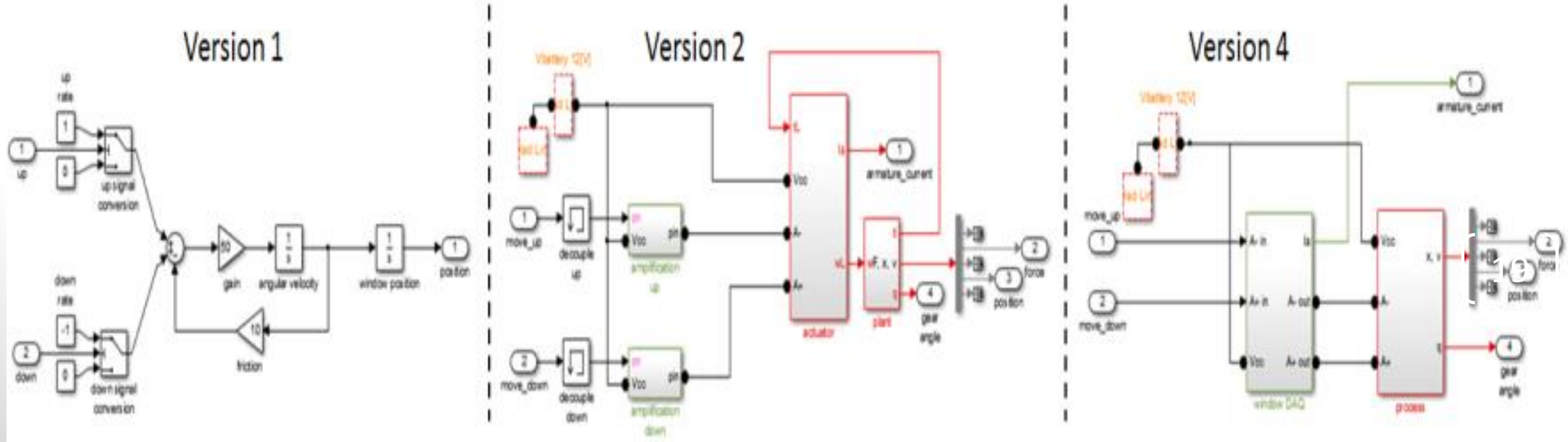
Detect_Obstacle
ID 6



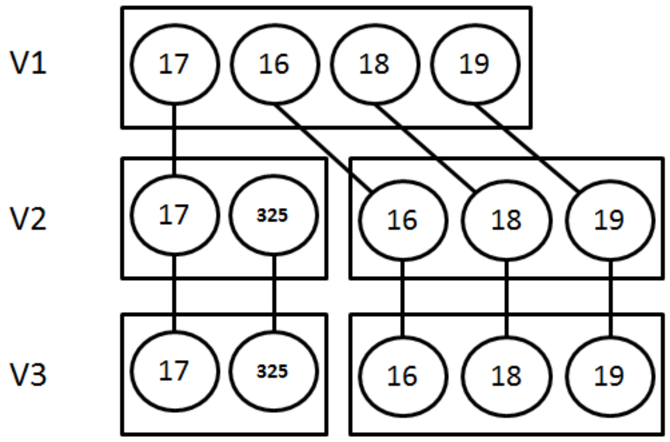
Power Window – MCC 2

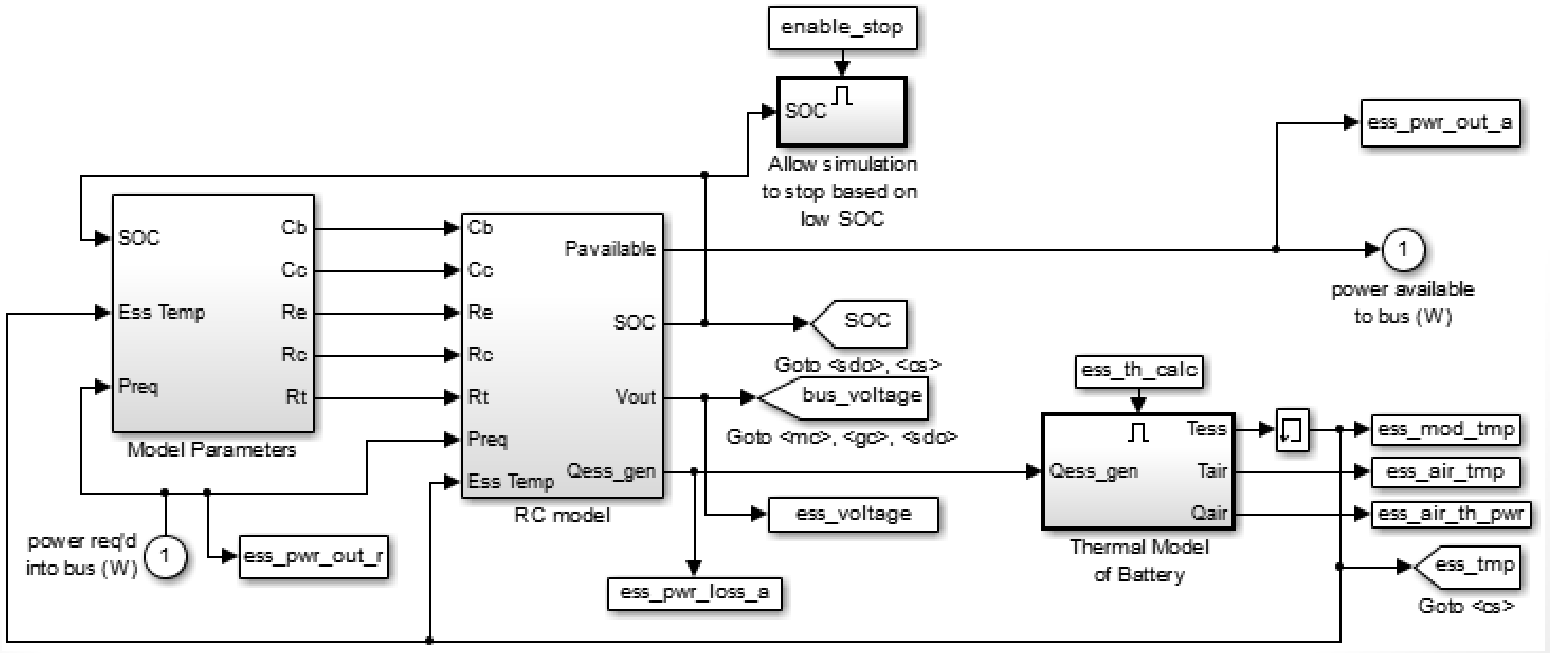


V5 Does not belong to any MCC



Advanced Vehicle Simulator – MCC7





Related Work

GCad

- Already discussed earlier

(Meta) Model Evolution Approaches

- To use for MCE -> Essentially have to create a system containing only the clones.

Model Comparison Approaches

- Surveyed the work (survey reference in paper)
- Nothing that explicates the structural evolution of Simulink

Simulink Refactoring

- Closet match is 1 paper on Simulink Refactoring by Matlab people.
- Related to antipatterns and refactoring steps

Future Work

Differencing and Visualization of Changes

- Incorporating it within SIMCCT -> Select an MCI and see how it changes from 1 version to the next.
- Integrate with our current work on Simulink Patterns

Other types of Models

- Currently working on MCD for Stateflow and behavioral models.
- Believe our work can be applied as long as they have concept of MCI and MCC.

Enumerating set of operations that affect MCE

- Plan on enumerating a set of Simulink model evolutions as they relate to model clone evolution
- Purpose: Find a sufficient set for performing MCD evaluation in a mutation-based framework. (thesis work)

Conclusions

- MCE Research is valuable -> Can help with ME
- Took first steps towards understanding Simulink MCE
- Developed SIMCCT to trace an MCC's MCIs across versions
- Executed SIMCCT on 3 Systems
- Provided and discussed examples
- Future improvement: Simulink Diff. and Vis. for a specific MCI

Questions?