MMINT-A: A Tool for Automated Change Impact Assessment on Assurance Cases

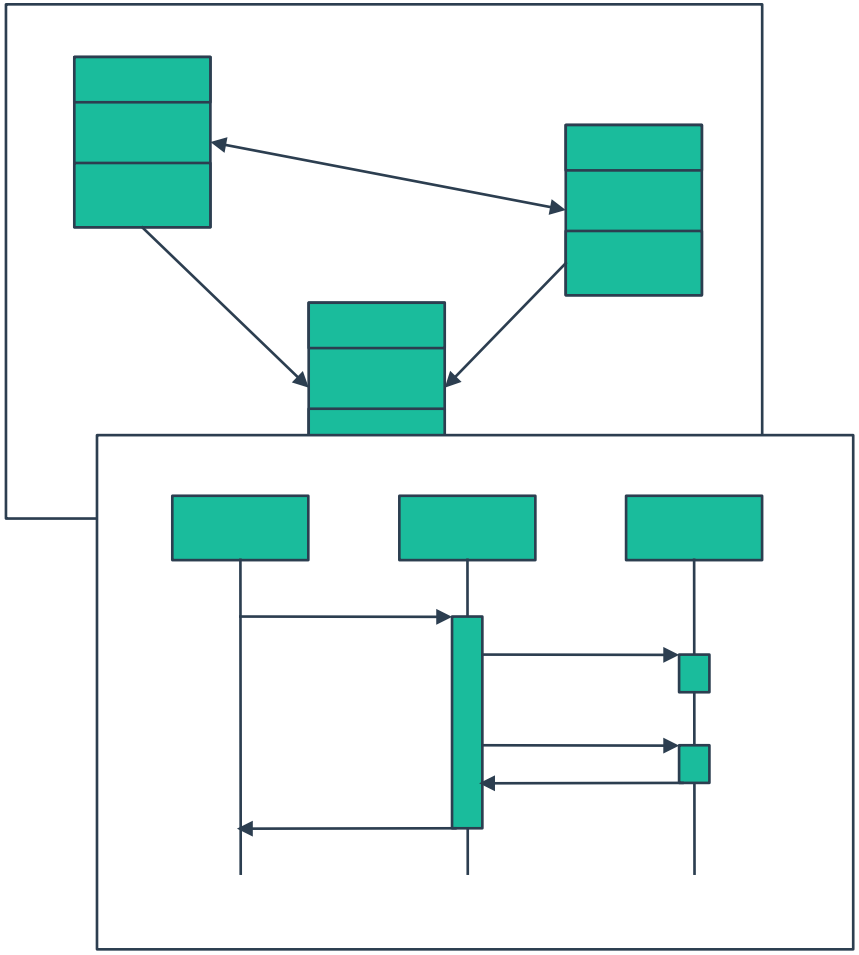
Nick Fung, Sahar Kokaly, Alessio Di Sandro,
Rick Salay, Marsha Chechik

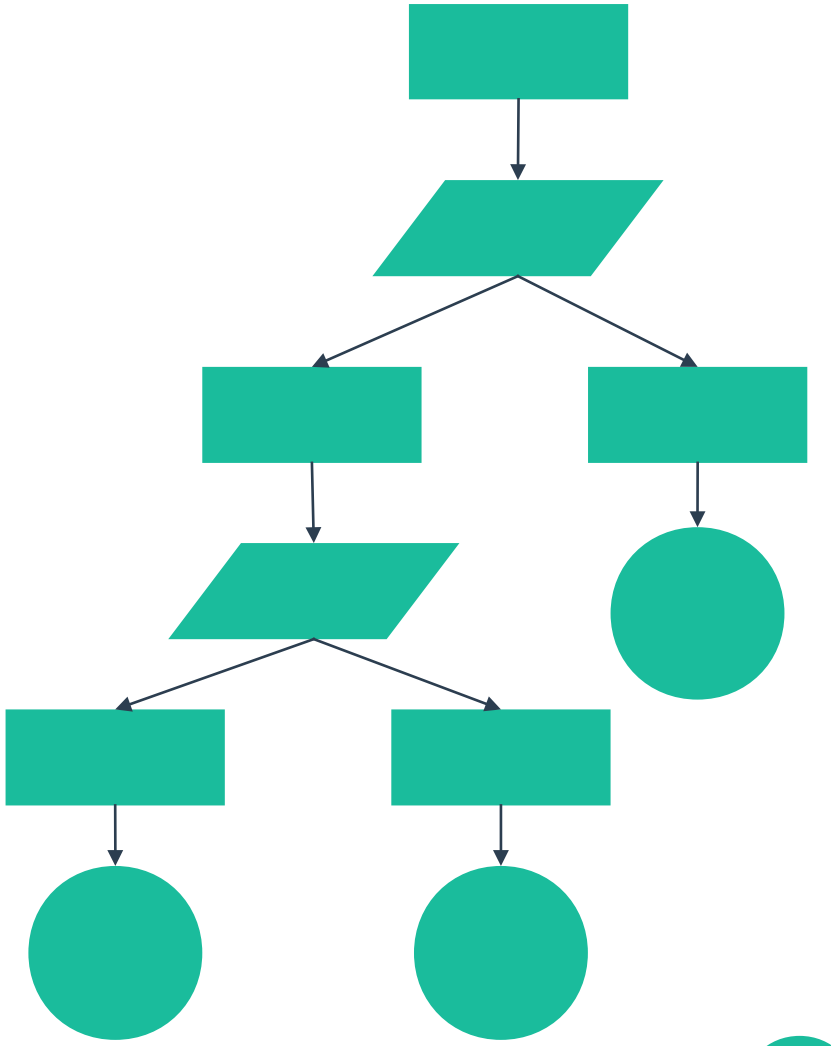
September 18, 2018

UNIVERSITY OF TORONTO

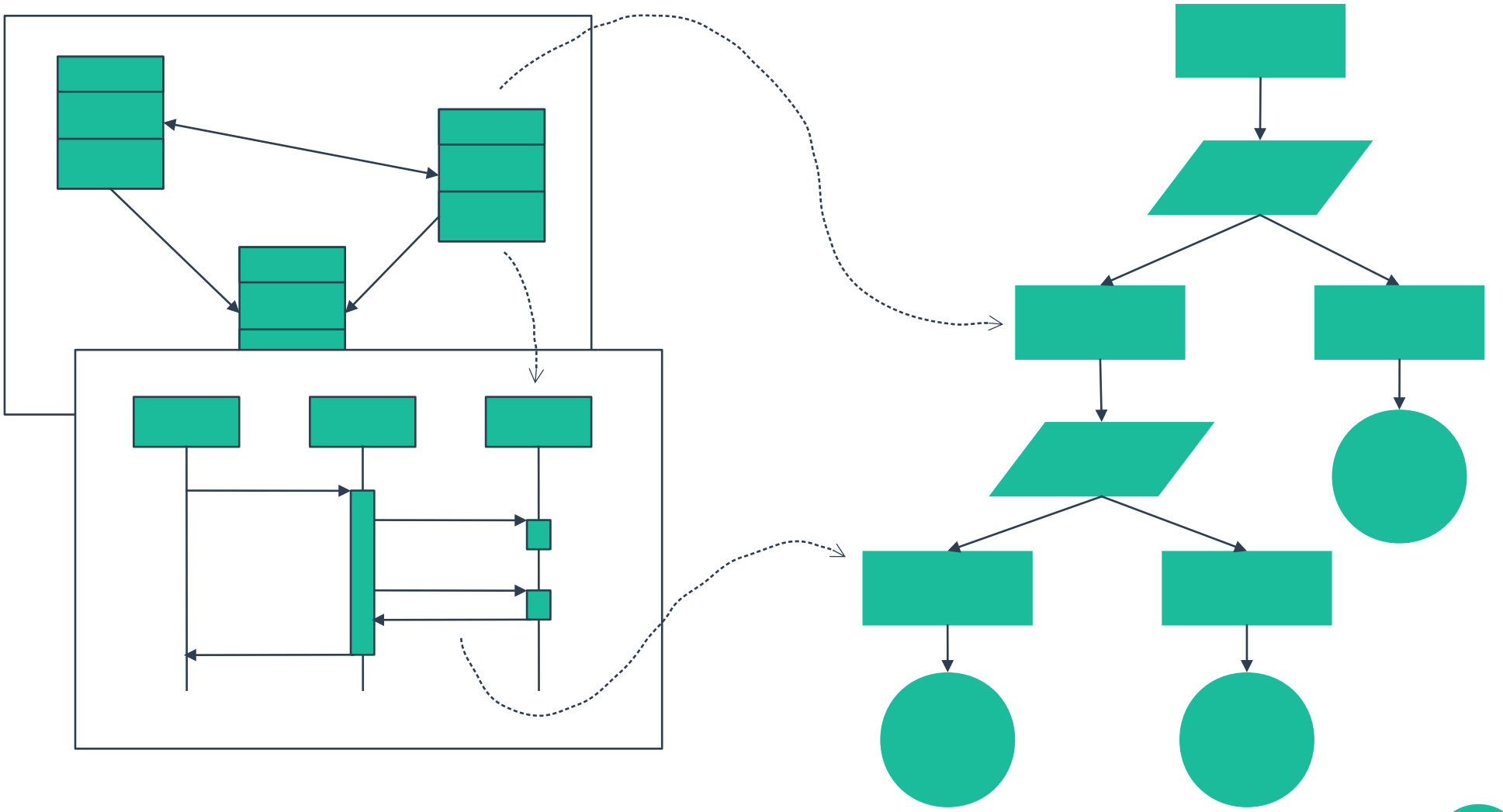
McMaster University

Motivation

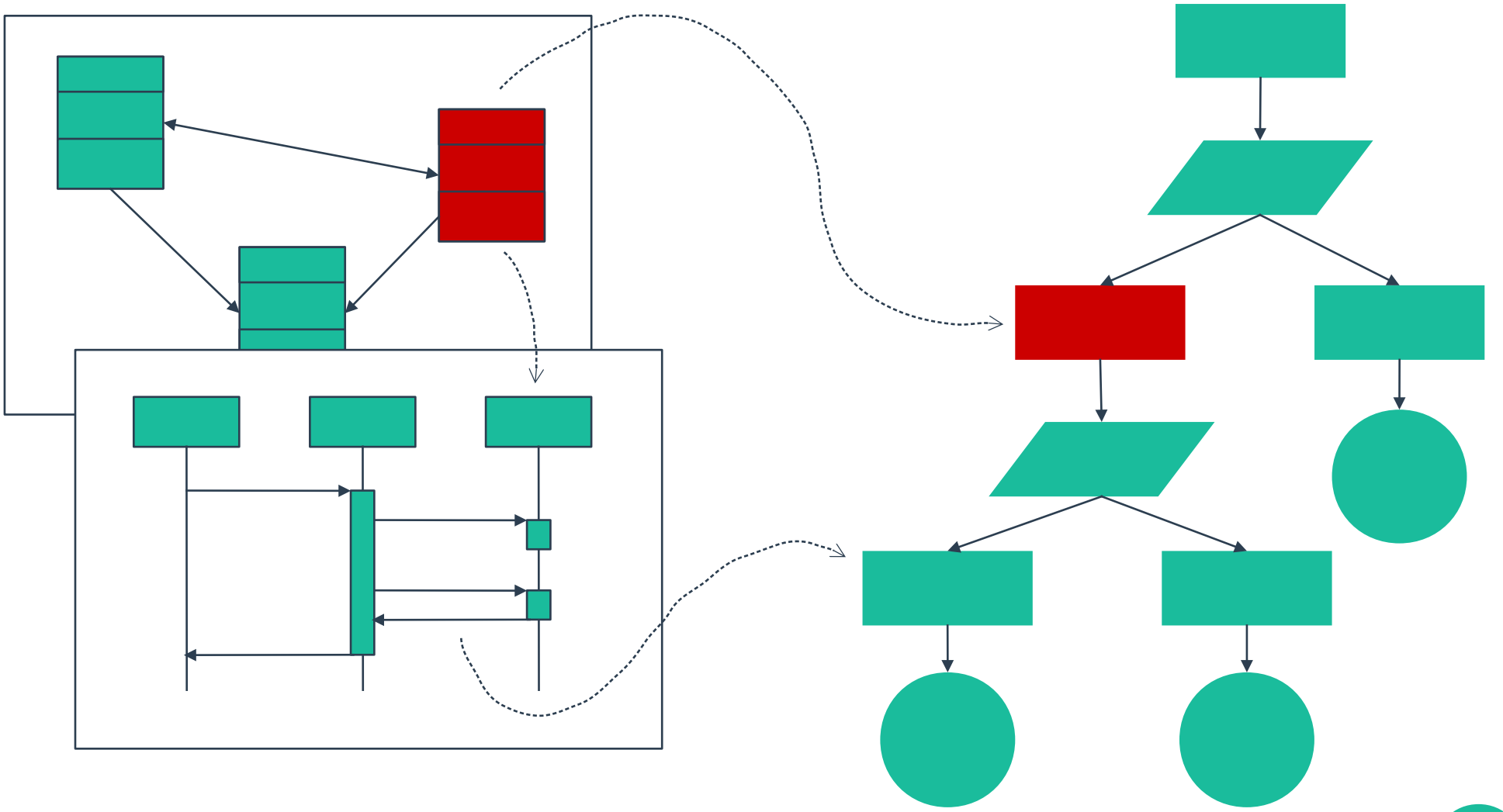




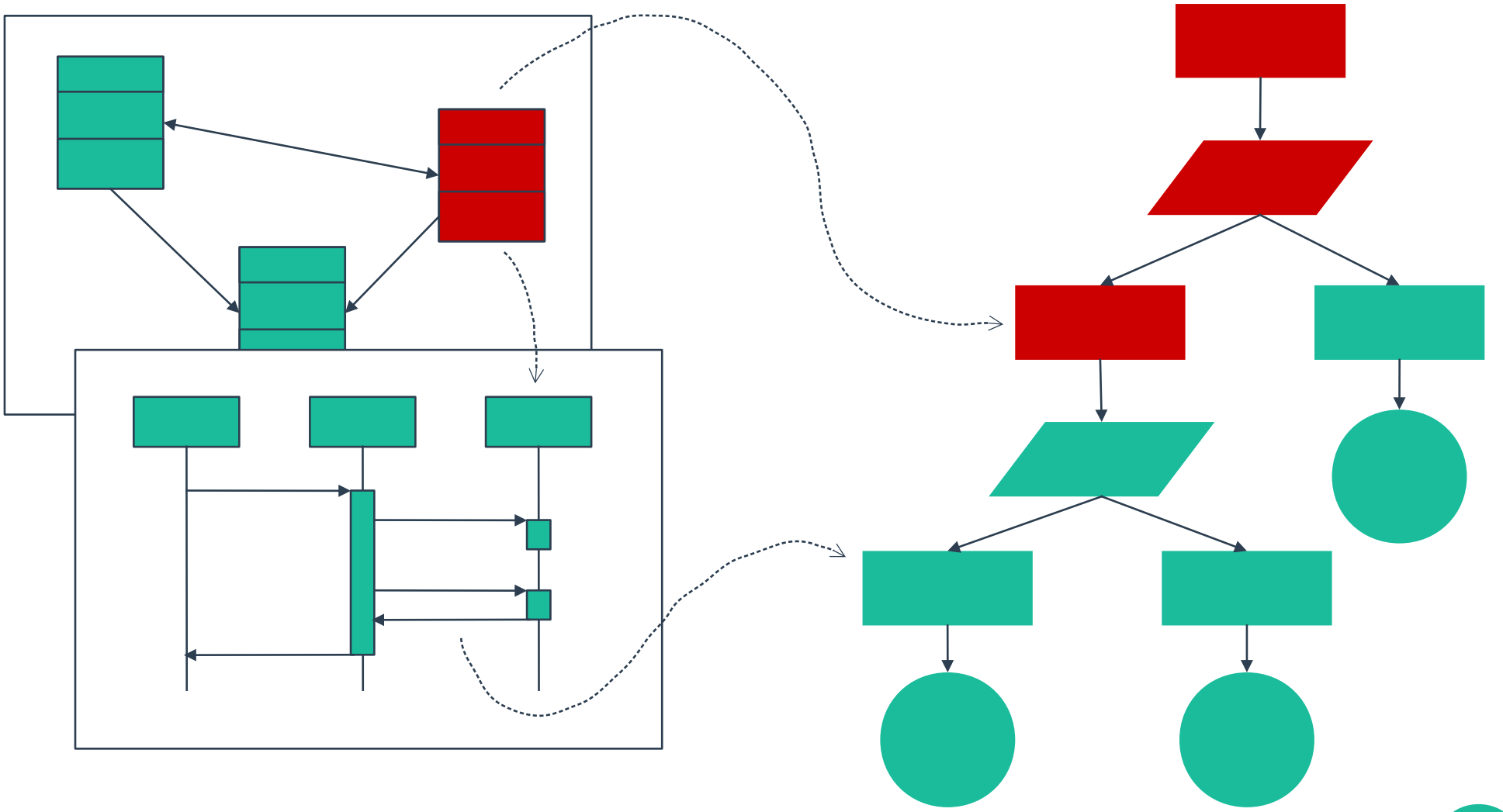
Motivation



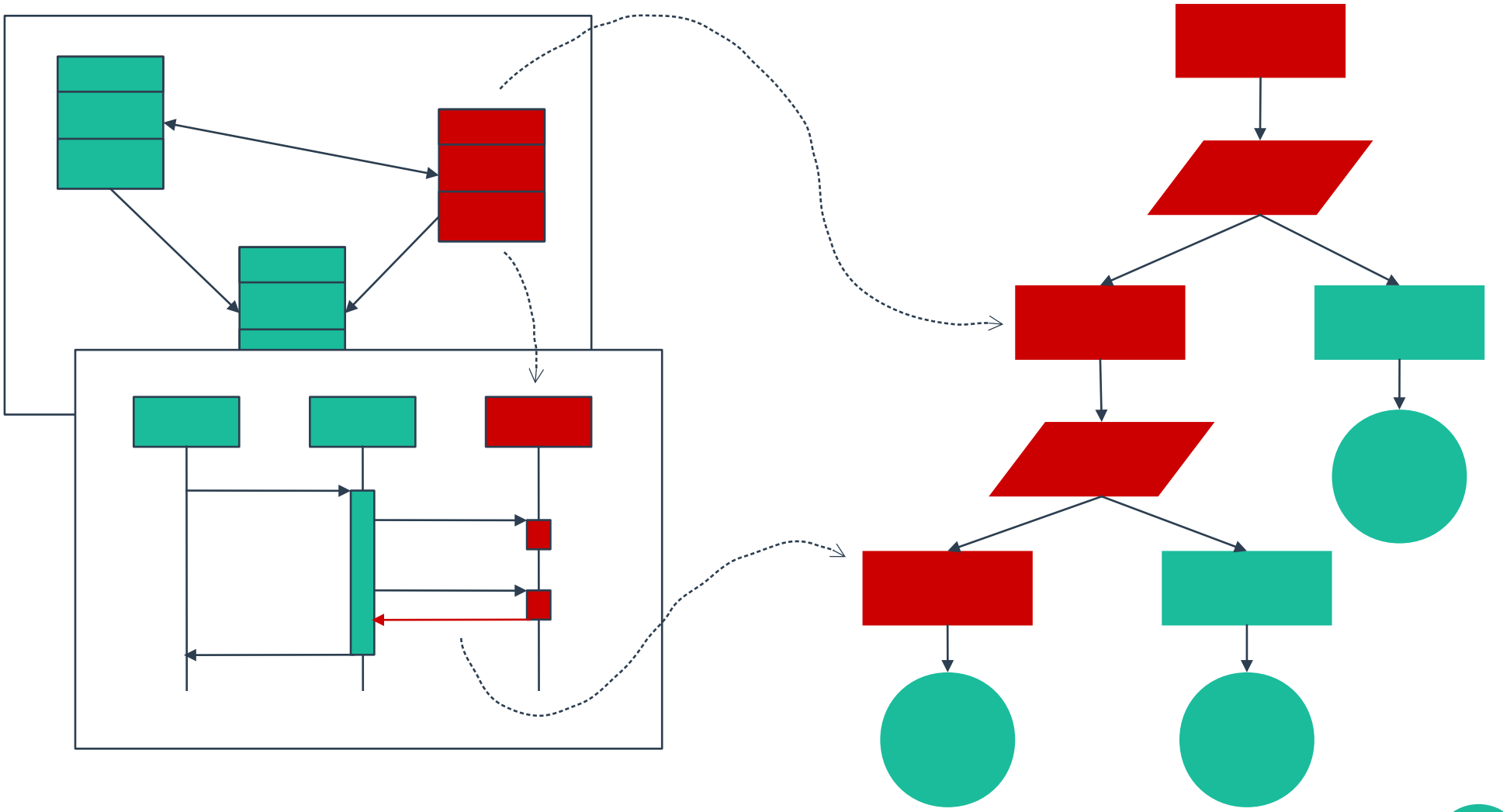
Motivation



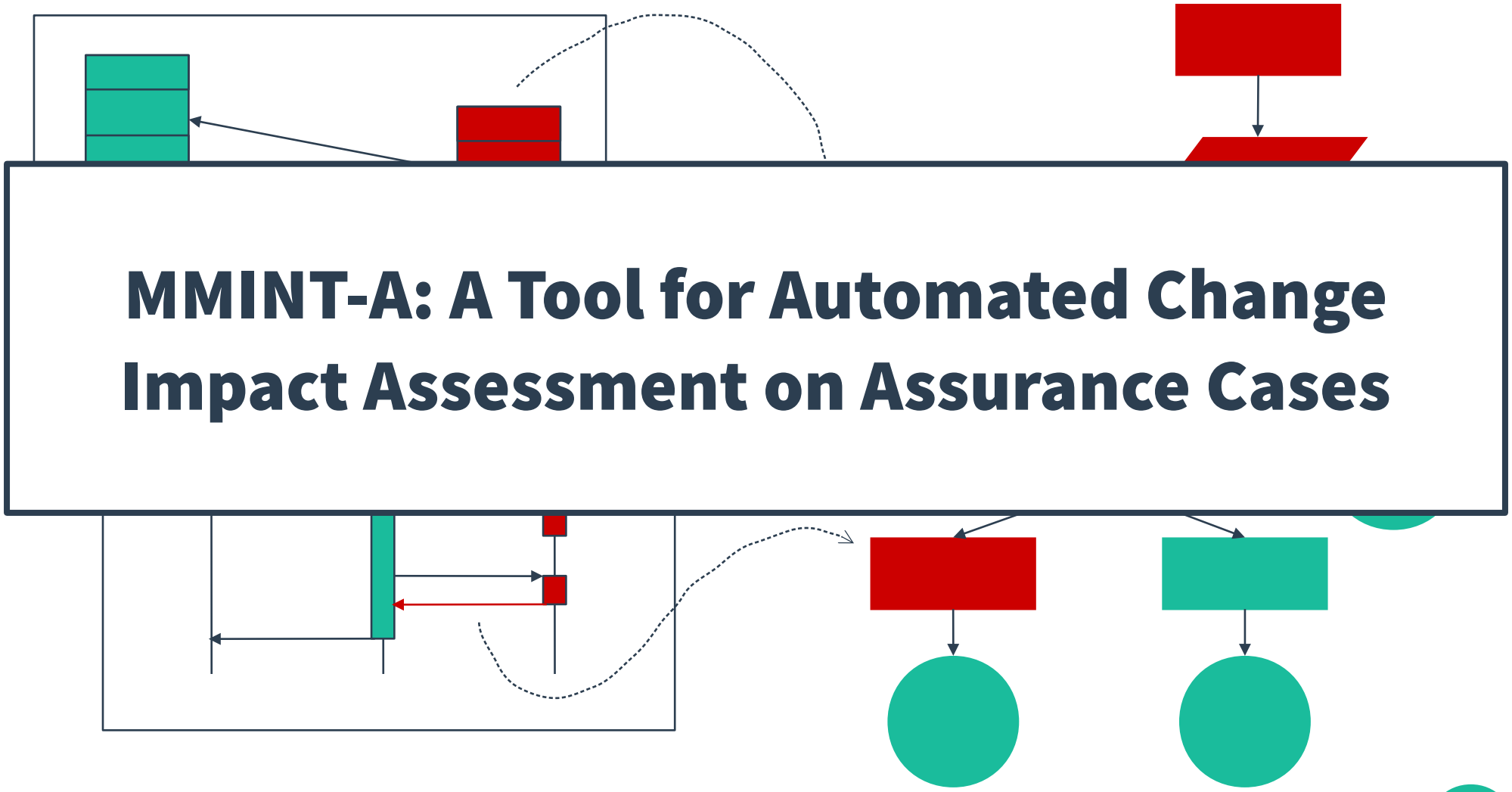
Motivation



Motivation

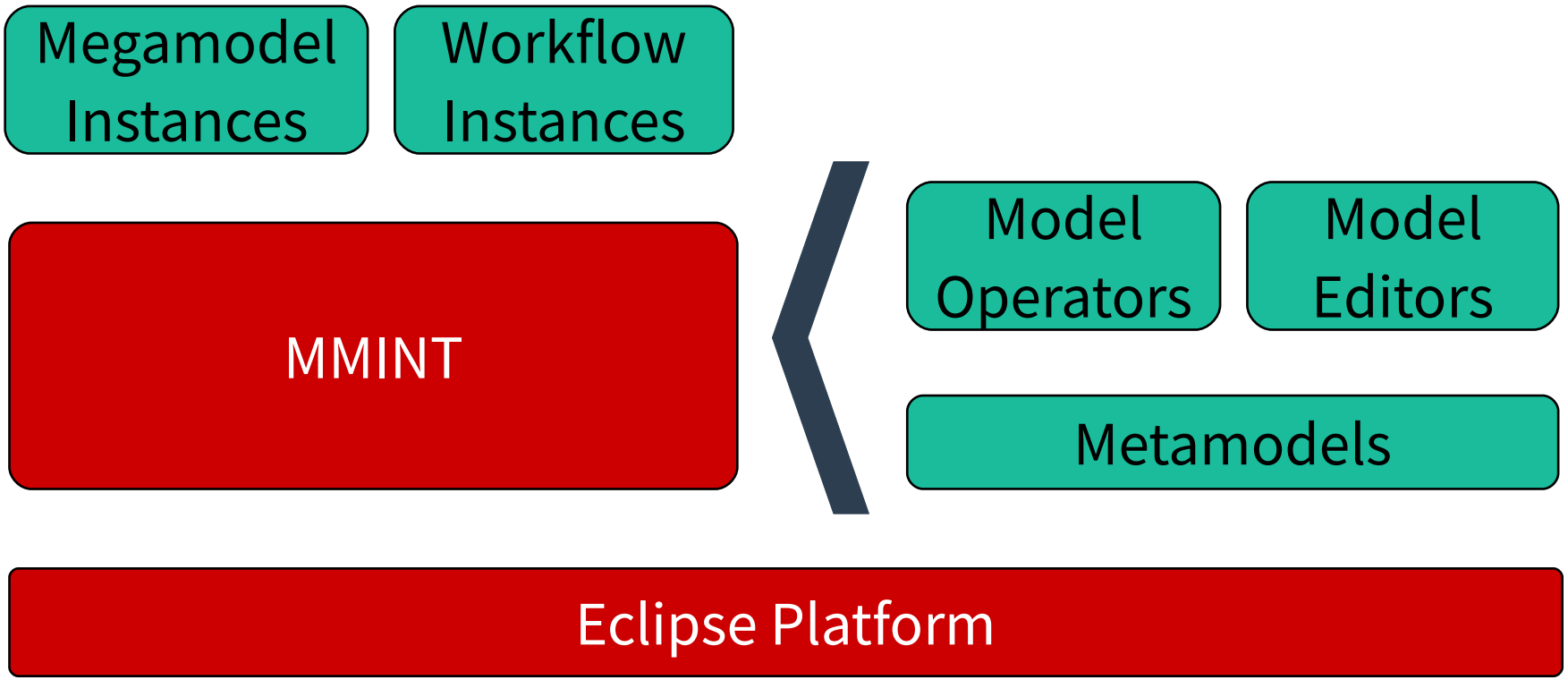


Motivation

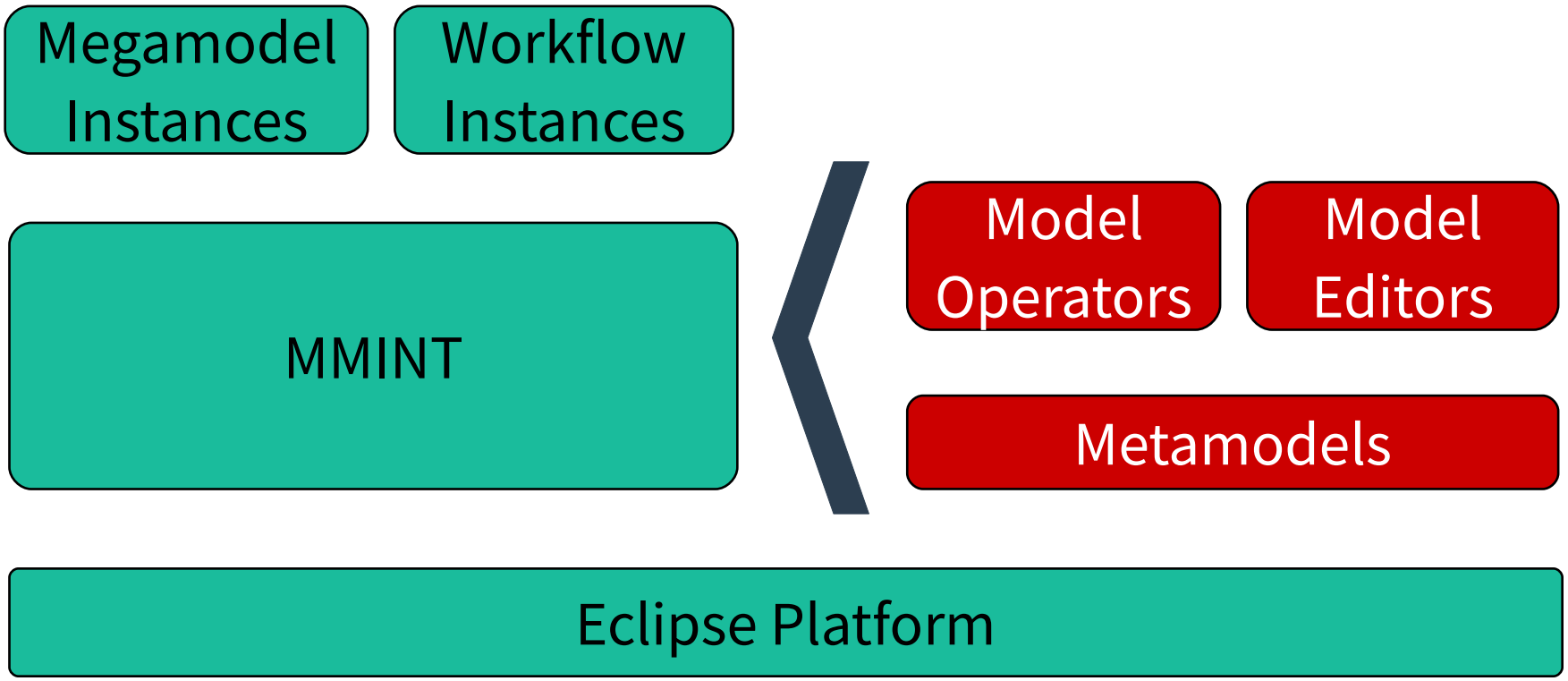


MMINT-A: A Tool for Automated Change Impact Assessment on Assurance Cases

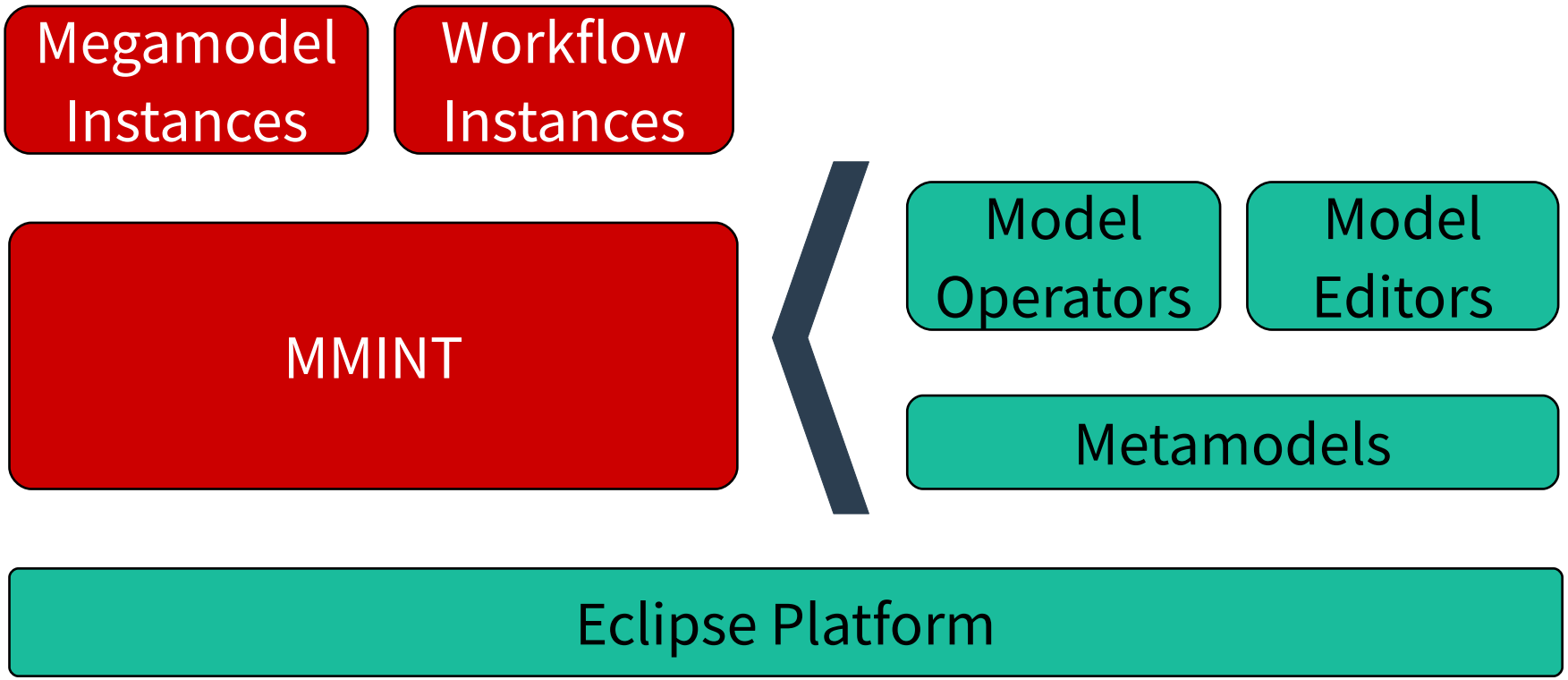
MMINT: Model Management Interactive



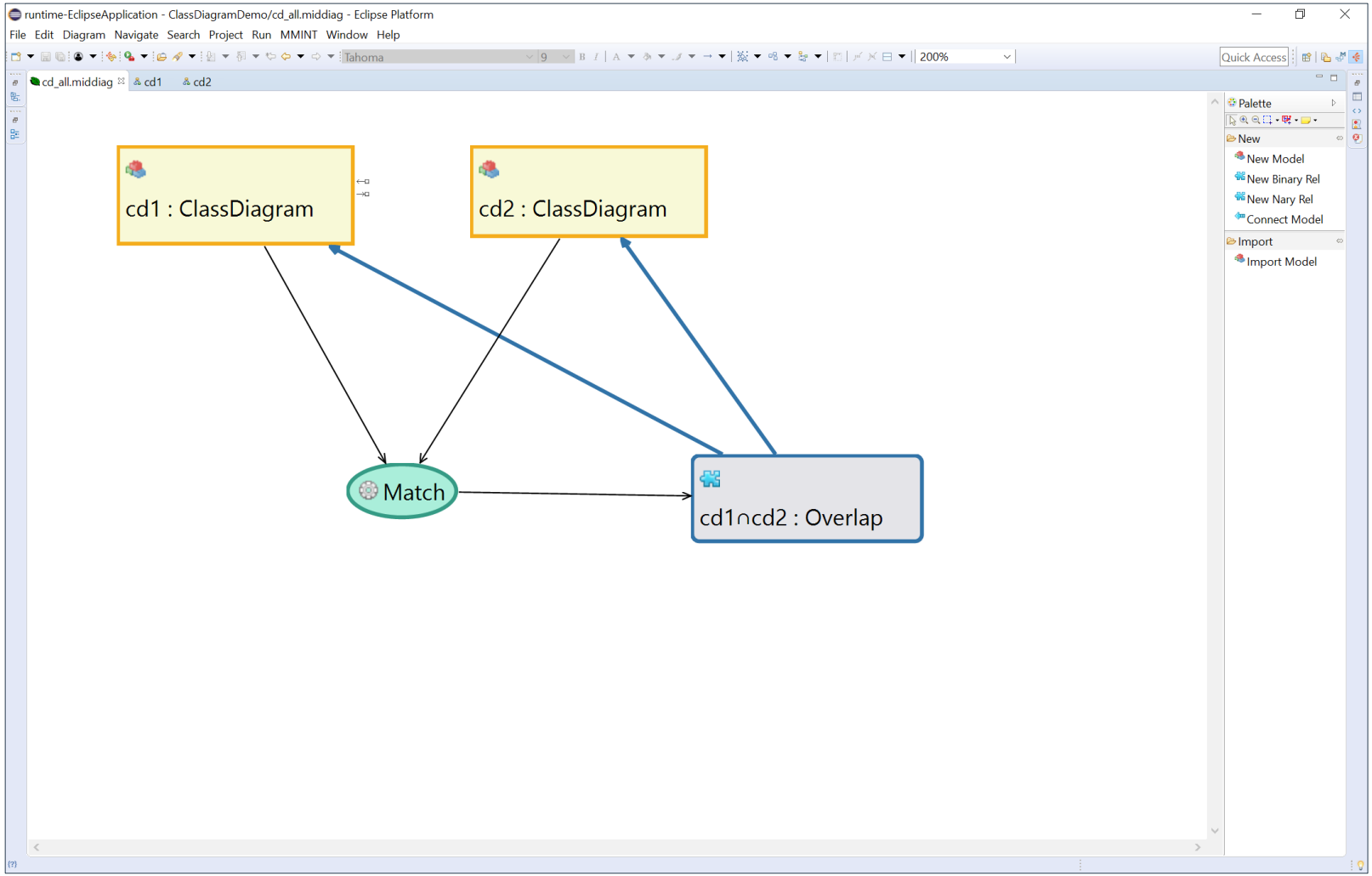
MMINT: Model Management Interactive



MMINT: Model Management Interactive



MMINT Demo


runtime-EclipseApplication - ClassDiagramDemo/cd_all.middiag - Eclipse Platform
File Edit Diagram Navigate Search Project Run MMINT Window Help
Quick Access
cd_all.middiag
cd1
cd2
Palette
New
New Model
New Binary Rel
New Nary Rel
Connect Model
Import
Import Model
cd1 : ClassDiagram
cd2 : ClassDiagram
Match
cd1∩cd2 : Overlap

Change Impact Assessment Algorithm

MODELS'16

A Model Management Approach for Assurance Case Reuse due to System Evolution

Sahar Kokaly
McMaster Centre for Software Certification
McMaster University
Hamilton, Canada
kokalys@mcmaster.ca

Rick Salay
Department of Computer Science
University of Toronto
Toronto, Canada
rsalay@cs.toronto.edu

Valentin Cassano
McMaster Centre for Software Certification
McMaster University
Hamilton, Canada
cassanv@mcmaster.ca

Tom Maibaum
McMaster Centre for Software Certification
McMaster University
Hamilton, Canada
maibaurn@mcmaster.ca

Marsha Chechik
Department of Computer Science
University of Toronto
Toronto, Canada
chechik@cs.toronto.edu

ABSTRACT

Evolution in software systems is a necessary activity that occurs due to fixing bugs, adding functionality or improving system quality. Systems often need to be shown to comply with regulatory standards. Along with demonstrating compliance, an artifact, called an assurance case, is often produced to show that the system indeed satisfies the property imposed by the standard (e.g., safety, privacy, security, etc.). Since each of the system, the standard, and the assurance case can be presented as a model, we propose the extension and use of traditional model management operators to aid in the reuse of parts of the assurance case when the system undergoes an evolution. Specifically, we present a model management approach that eventually produces a partial evolved assurance case and guidelines to help the assurance engineer in completing it. We demonstrate how our approach works on an automotive subsystem regulated by the ISO 26262 standard.

Keywords

Evolution, reuse, model management, regulatory compliance, assurance cases, certification.

1. INTRODUCTION

The pervasiveness of software in all aspects of human activity has created special concerns regarding issues such as safety, security and privacy. Governments and standard organizations (e.g., ISO) have responded to this trend by creating regulations and standards that software must comply with. For companies, compliance is a complex and costly goal to achieve. They may have to comply with multiple standards due to multiple jurisdictions and track the changes in standards. Assurance cases – collections of arguments and evidence to support the claims of compliance – must be developed and managed. Finally, maintaining families of related software products further multiplies the effort. Increasingly, models and model-driven engineering are being used as means to facilitate communication and collaboration between the stakeholders in the compliance and safety chain and further to introduce automation into regulatory compliance tasks.

In a position paper [21], we laid out a research agenda for applying model management to address the software compliance problem and sketched its use in particular compliance management scenarios. In this paper, we focus on one of these scenarios – assurance case reuse due to system evolution – and develop it in detail. Fig. 1 illustrates the scenario at a high level. Assume that a current specification S describes the specification for the software in a vehicle. In addition, a type of assurance case A, called a *safety case*, has been developed complying with the ISO 26262 vehicle functional safety standard [16]. Safety case A contains perhaps thousands of safety claims about different components of the vehicle, as well as arguments and evidence to support these claims. Now if S is evolved to S' – for example, as a result of a new requirement or a bug fix – a corresponding safety case A' for S' must be developed. Due to complexity and effort required to develop a safety case, there is strong incentive to reuse as much of A as possible in the creation of A'. We address this problem using a model management strategy. Specifically, we make the following contributions:

1. We define a generic model management framework for assurance case reuse due to model evolution.
2. We identify and specify the model management operators needed for a semi-automated solution to the assurance case reuse problem and present an algorithm for reuse.
3. We evaluate the generic framework and process of solution by instantiating it for ISO 26262 vehicle safety cases with the KAOS goal modeling language [6] used in existing assurance cases. We then apply this instantiation to an automotive subsystem, namely, a power sliding

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MODELS '16, October 02-07, 2016, Saint-Malo, France

© 2016 ACM. ISBN 978-1-4503-4321-3/16/10…$15.00

DOI: http://dx.doi.org/10.1145/2976767.2976792

SafeComp'17

Safety Case Impact Assessment in Automotive Software Systems: An Improved Model-Based Approach

Sahar Kokaly[1(✉)], Rick Salay[2], Marsha Chechik[2], Mark Lawford[1], and Tom Maibaum[1]

[1] McMaster Centre for Software Certification, McMaster University, Hamilton, Canada
{kokalys,lawford,maibaum}@mcmaster.ca

[2] Department of Computer Science, University of Toronto, Toronto, Canada
{rsalay,chechik}@cs.toronto.edu

Abstract. Like most systems, automotive software systems evolve due to many reasons including adding, removing or modifying features, fixing bugs, or improving system quality. In this context, safety cases, used to demonstrate that a system satisfies predefined safety requirements, often dictated by a standard such as ISO 26262, need to co-evolve. A necessary step is performing an impact assessment to identify how changes in the system affect the safety case. In previous work, we introduced a generic model-based impact assessment approach, that, while sound, was not particularly precise. In this work, we show how exploiting knowledge about system changes, the particular safety case language, and the standard can increase the precision of the impact assessment, reducing any unnecessary revision work required by a safety engineer. We present six precision improvement techniques illustrated on a GSN safety case used with ISO 26262.

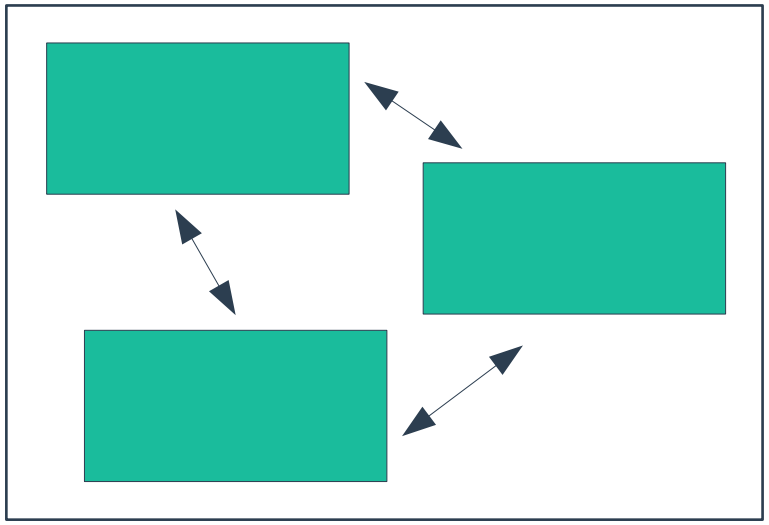
1 Introduction

Safety engineers in various domains, including automotive, experience difficulties with safety case maintenance. As stated in [11], the main reason for this is that they do not have a systematic approach by which to examine the impact of change on a safety argument. The authors of [2] performed a study which suggested that engineers spend 50–100 h on Change Impact Assessment (CIA) per year on average. The second most commonly mentioned CIA challenge is related to information overload. The three most senior engineers in the study reported that obtaining a system understanding is hard due to the complexity of the systems. The sheer number of software artifacts involved makes traceability information highly complex. Based on the results of [2], determining how a change impacts the product source code seems to be less of a challenge than determining impact on non-code artifacts, e.g., requirements, specifications, and test cases. In [14,17], the authors further discuss the problem of CIA being a

© Springer International Publishing AG 2017

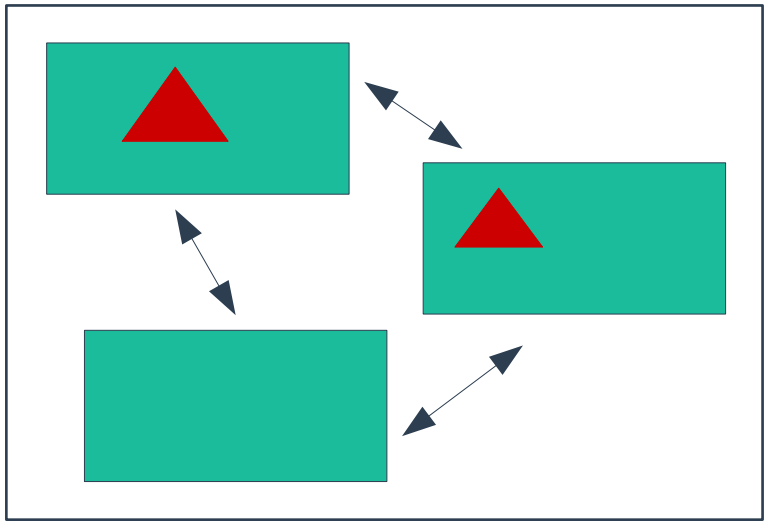
S. Tonetta et al. (Eds.): SAFECOMP 2017, LNCS 10488, pp. 69–85, 2017.
DOI: 10.1007/978-3-319-66266-4_5

Change Impact Assessment Inputs



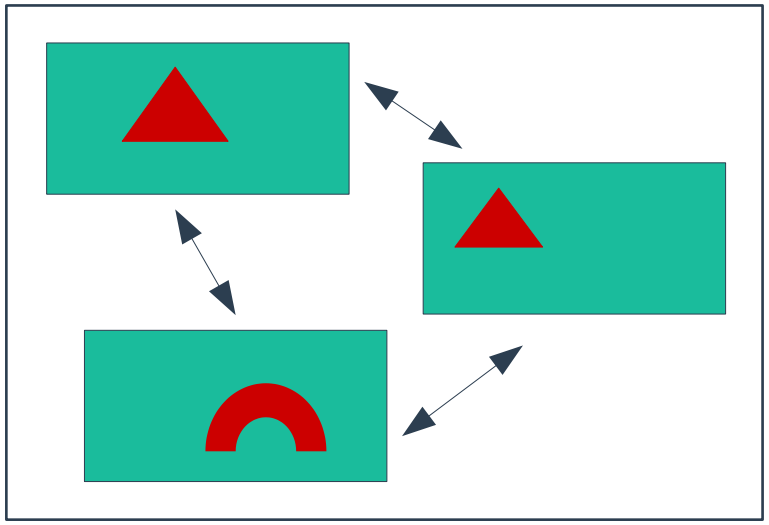
- System megamodel
- Set of modifications
- Set of deletions
- Assurance case megamodel

Change Impact Assessment Inputs

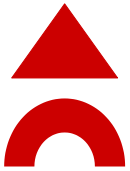


- System megamodel
- Set of modifications
- Set of deletions
- Assurance case megamodel

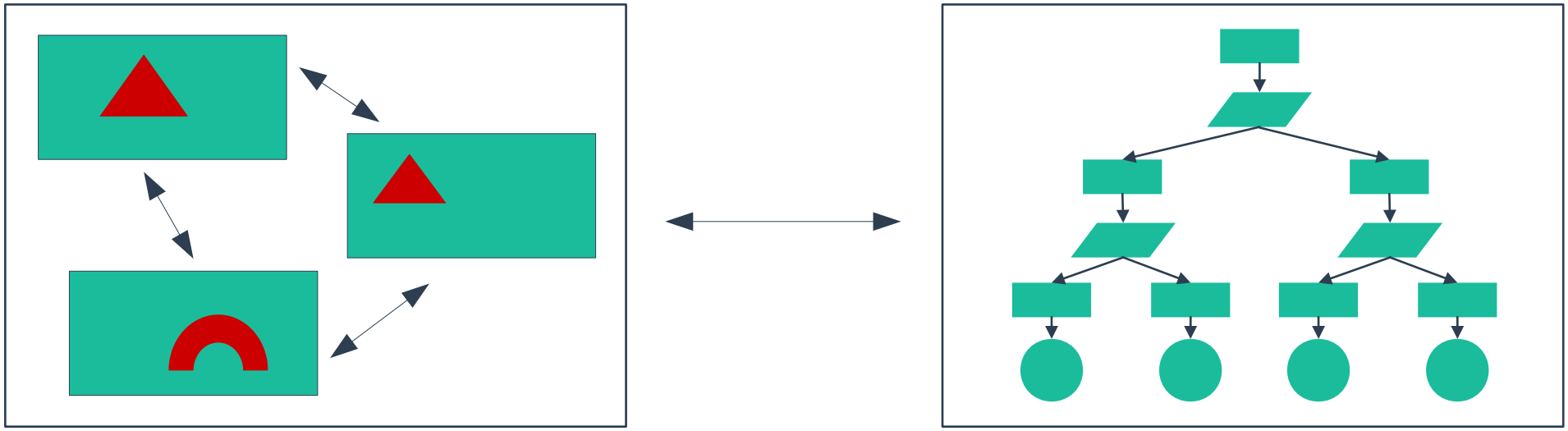
Change Impact Assessment Inputs



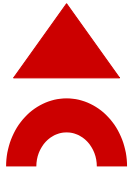
- System megamodel
- Set of modifications
- Set of deletions
- Assurance case megamodel



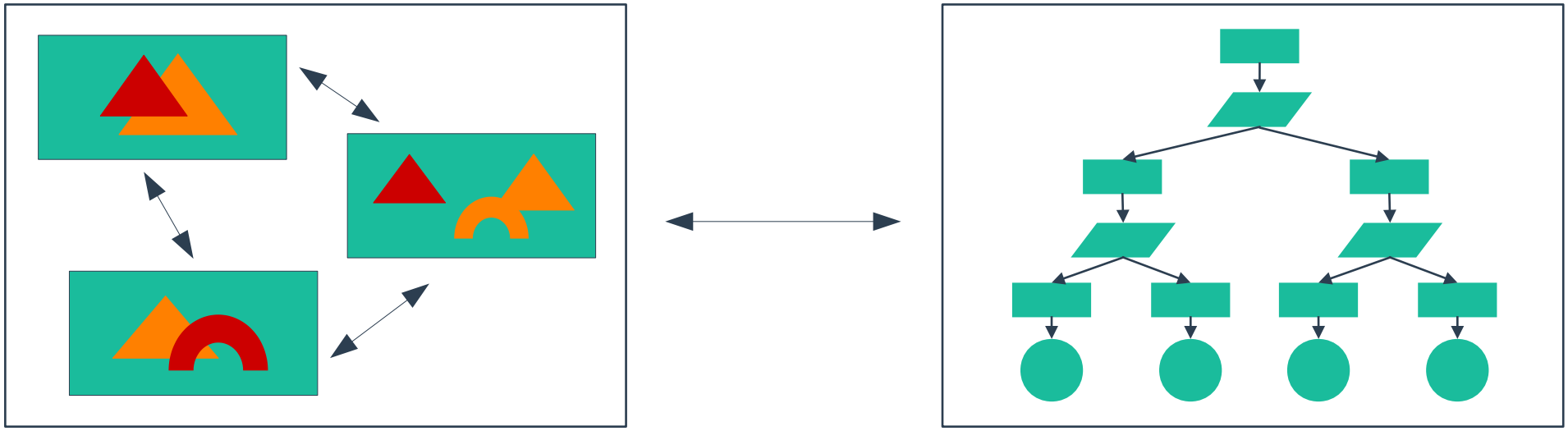
Change Impact Assessment Inputs



- System megamodel
- Set of modifications
- Set of deletions
- Assurance case megamodel

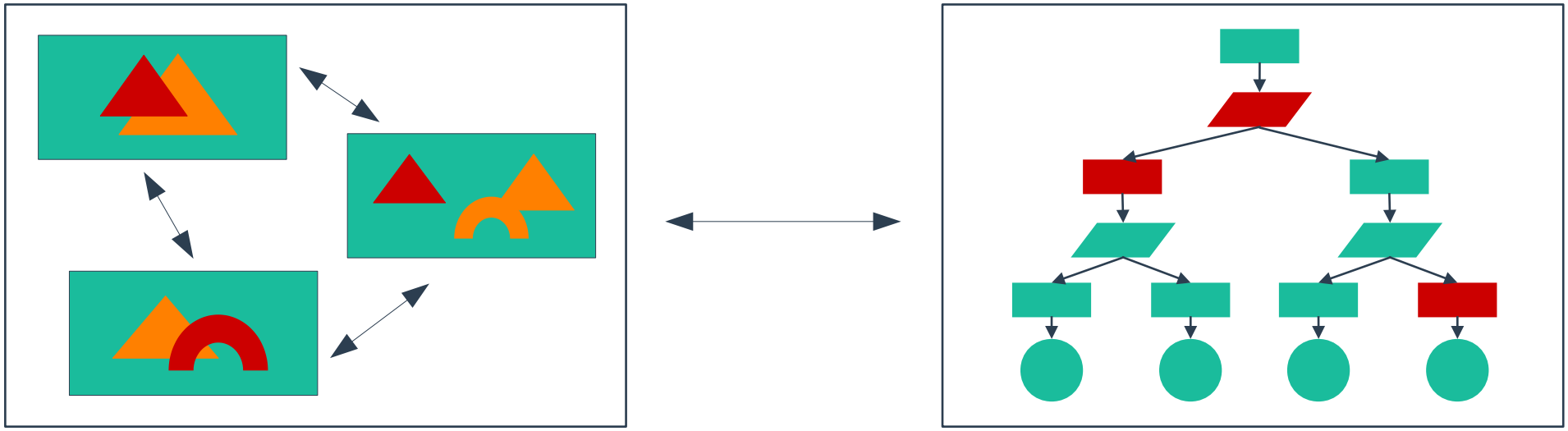


Change Impact Assessment Algorithm



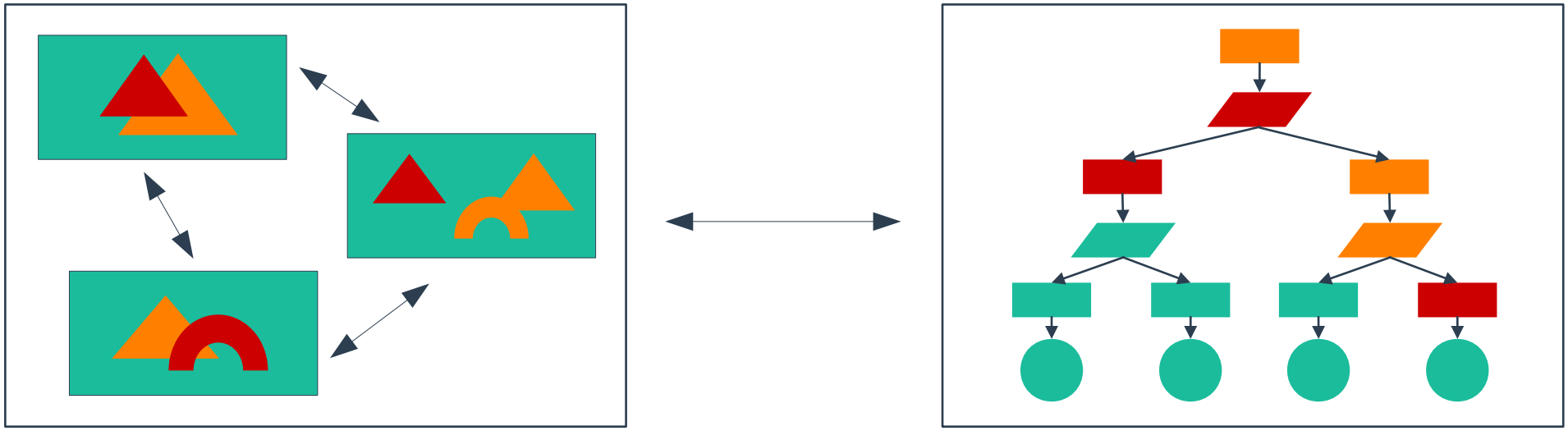
1) Slice system megamodel
2) Propagate impact to assurance case
3) Slice assurance case
4) Annotate assurance case

Change Impact Assessment Algorithm



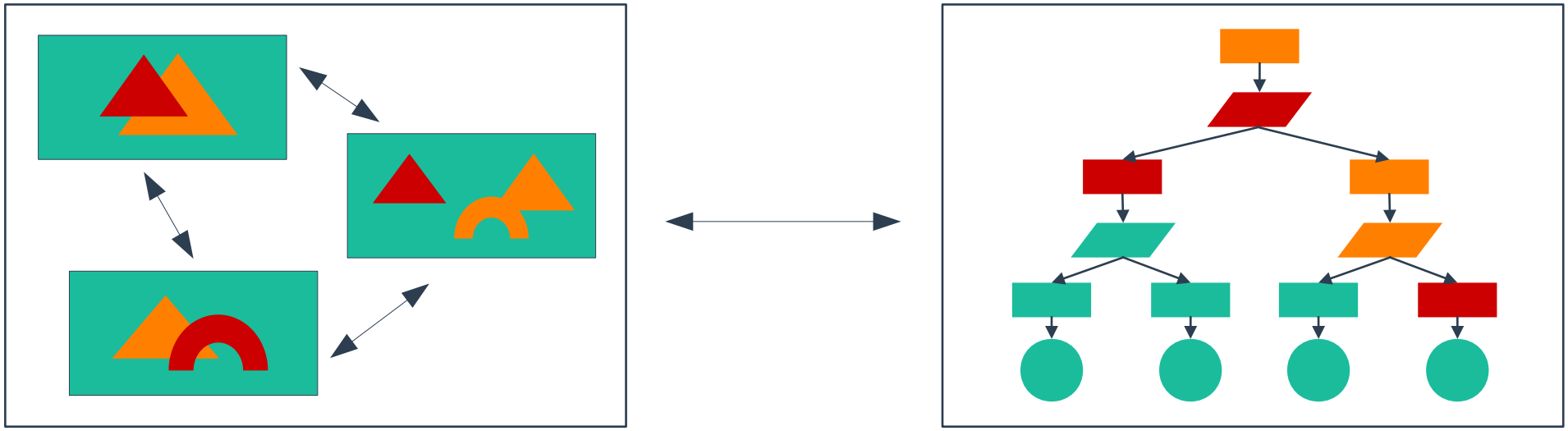
1) Slice system megamodel
2) Propagate impact to assurance case
3) Slice assurance case
4) Annotate assurance case

Change Impact Assessment Algorithm



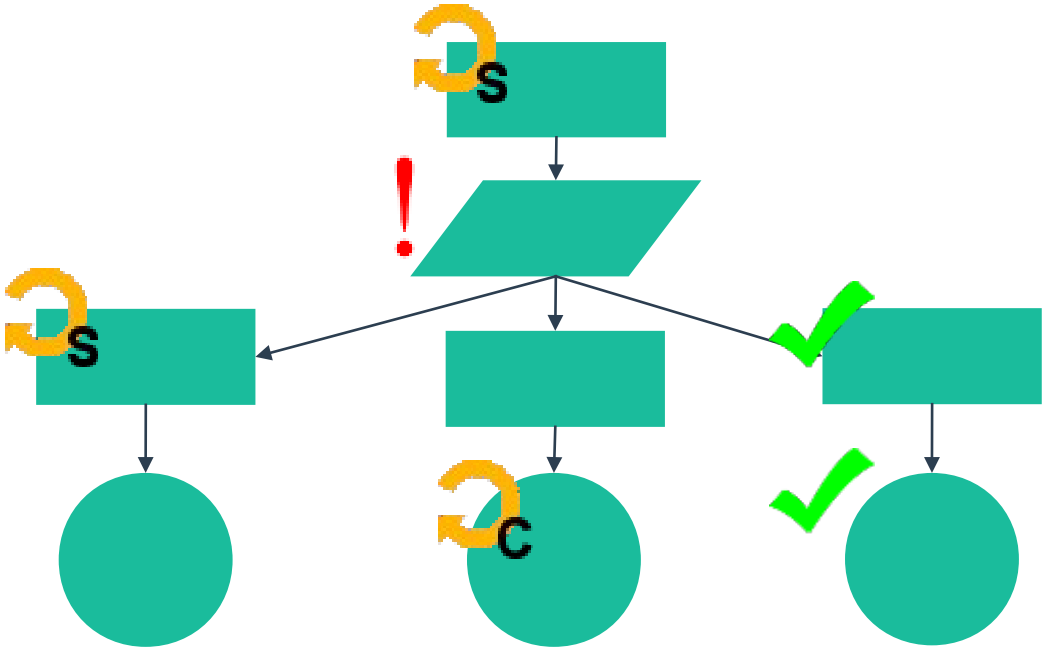
1) Slice system megamodel
2) Propagate impact to assurance case
3) Slice assurance case
4) Annotate assurance case

Change Impact Assessment Algorithm



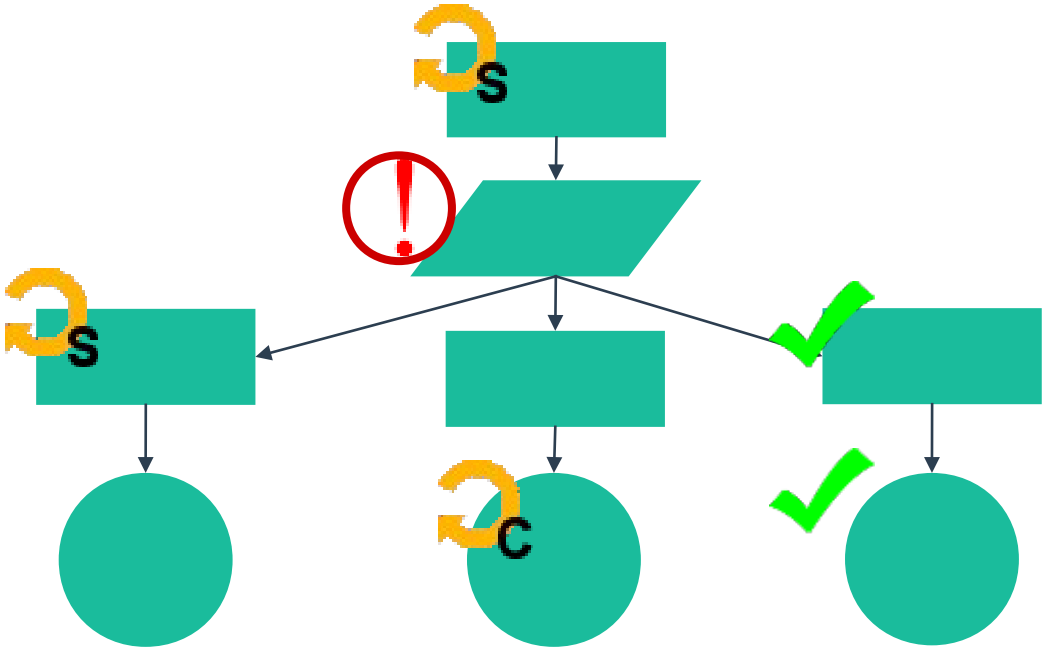
1) Slice system megamodel
2) Propagate impact to assurance case
3) Slice assurance case
4) Annotate assurance case

Change Impact Annotations



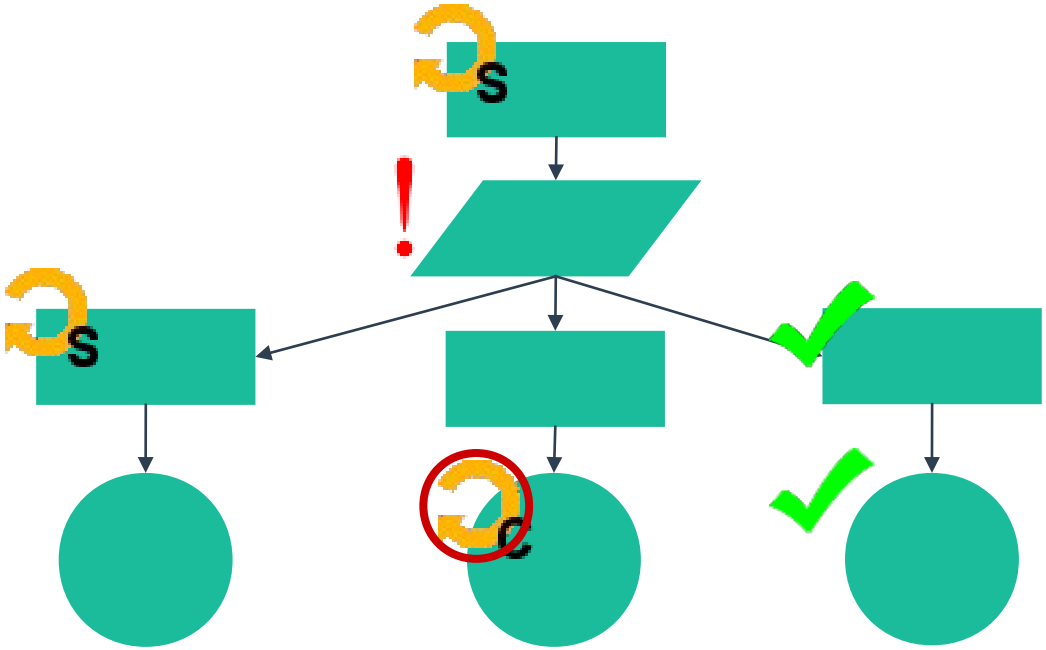
- **Revise:** Contents require changing
- **Recheck Content:** Contents may require changing
- **Recheck State:** Contents may not be supported
- **Reuse:** Contents are not supported

Change Impact Annotations



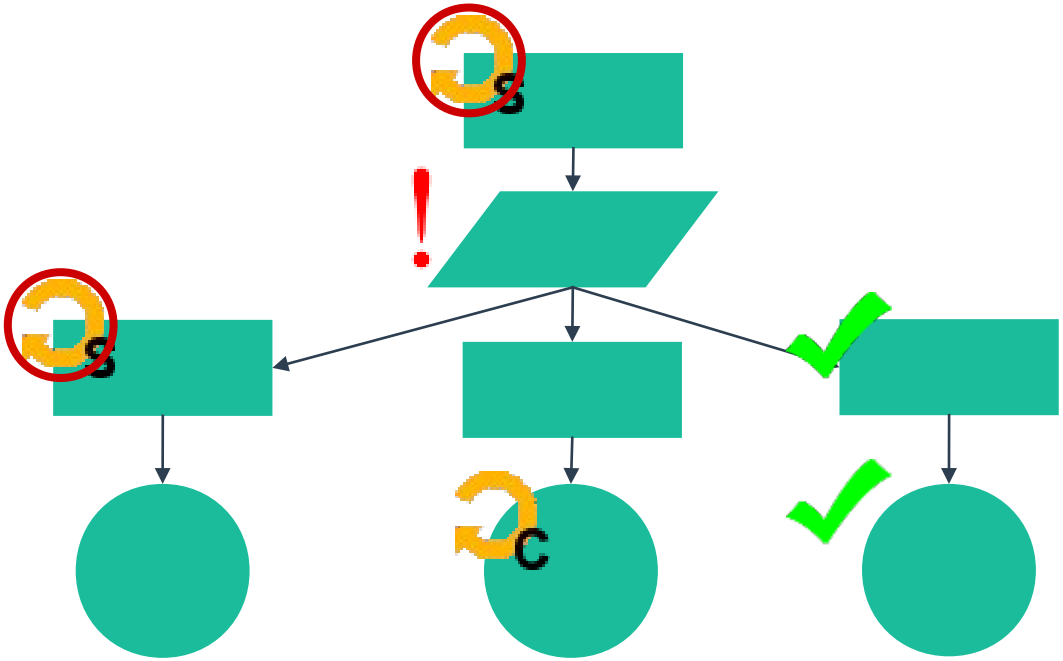
- **Revise:** Contents require changing
- **Recheck Content:** Contents may require changing
- **Recheck State:** Contents may not be supported
- **Reuse:** Contents are not supported

Change Impact Annotations



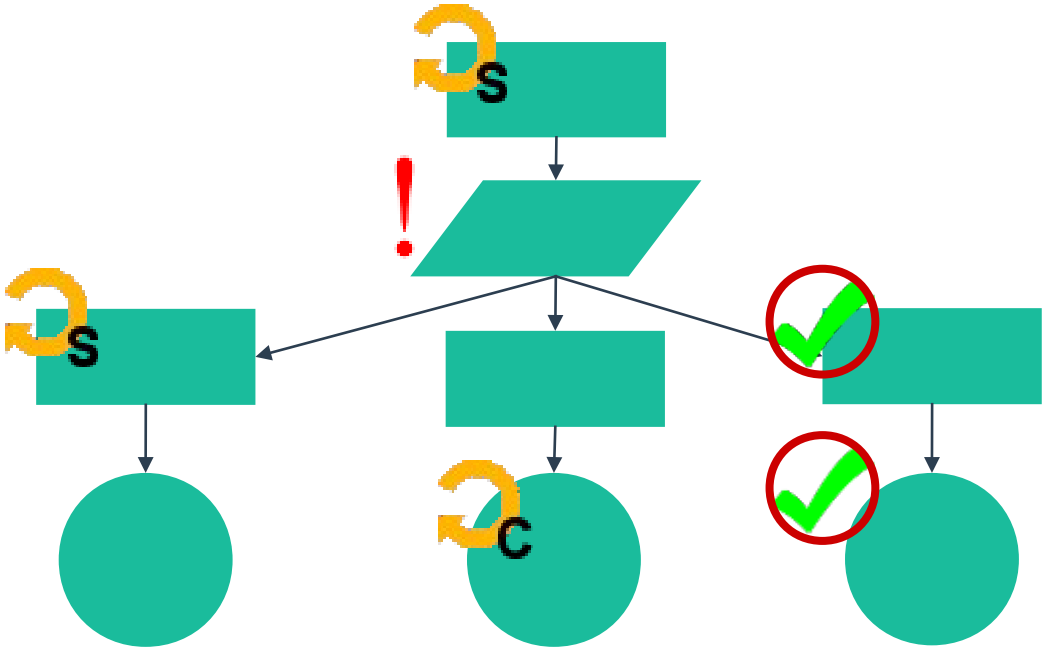
- **Revise:** Contents require changing
- **Recheck Content:** Contents may require changing
- **Recheck State:** Contents may not be supported
- **Reuse:** Contents are not supported

Change Impact Annotations



- **Revise:** Contents require changing
- **Recheck Content:** Contents may require changing
- **Recheck State:** Contents may not be supported
- Reuse: Contents are not supported

Change Impact Annotations

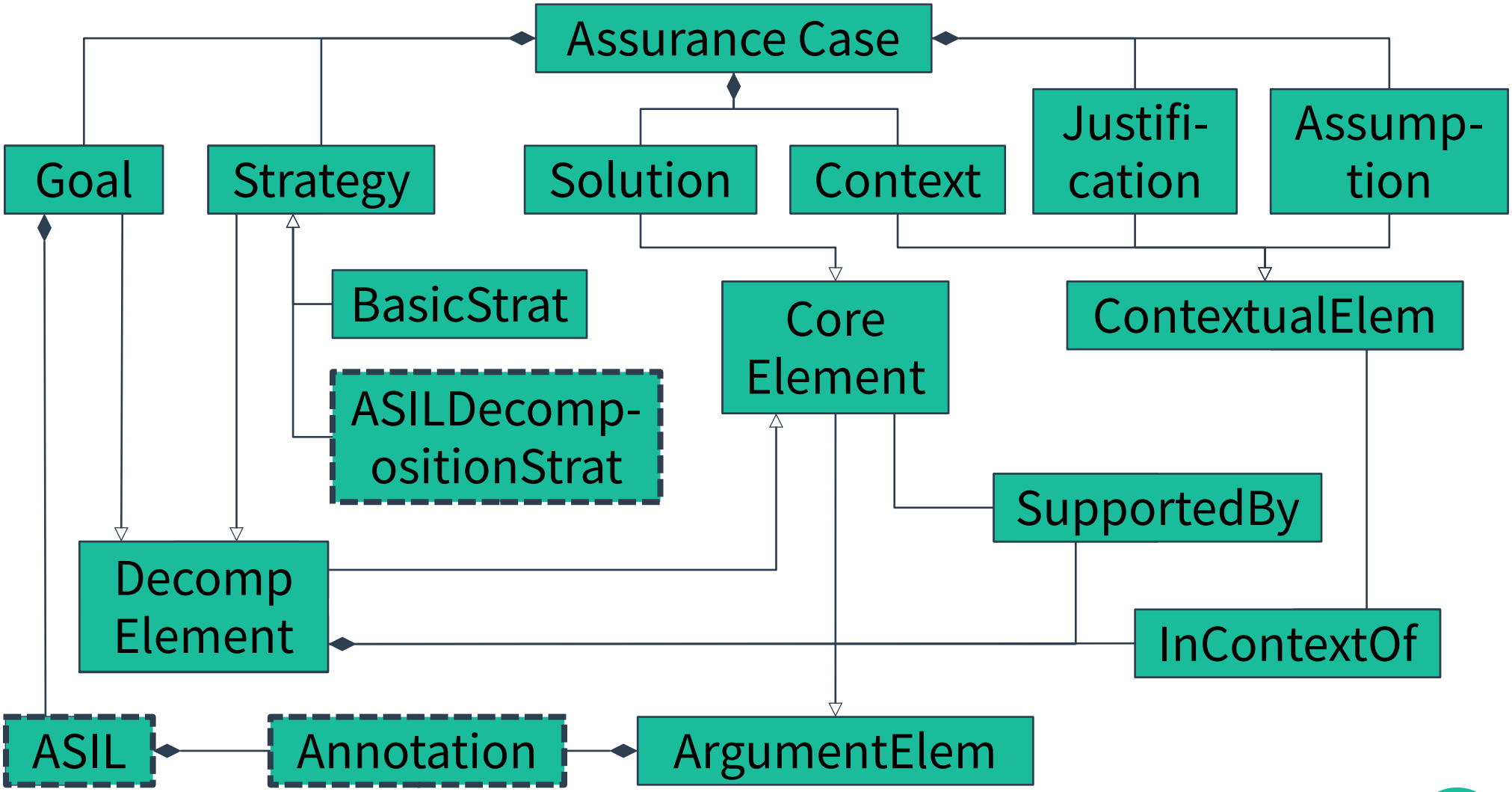


- **Revise:** Contents require changing
- **Recheck Content:** Contents may require changing
- **Recheck State:** Contents may not be supported
- **Reuse:** Contents are not supported

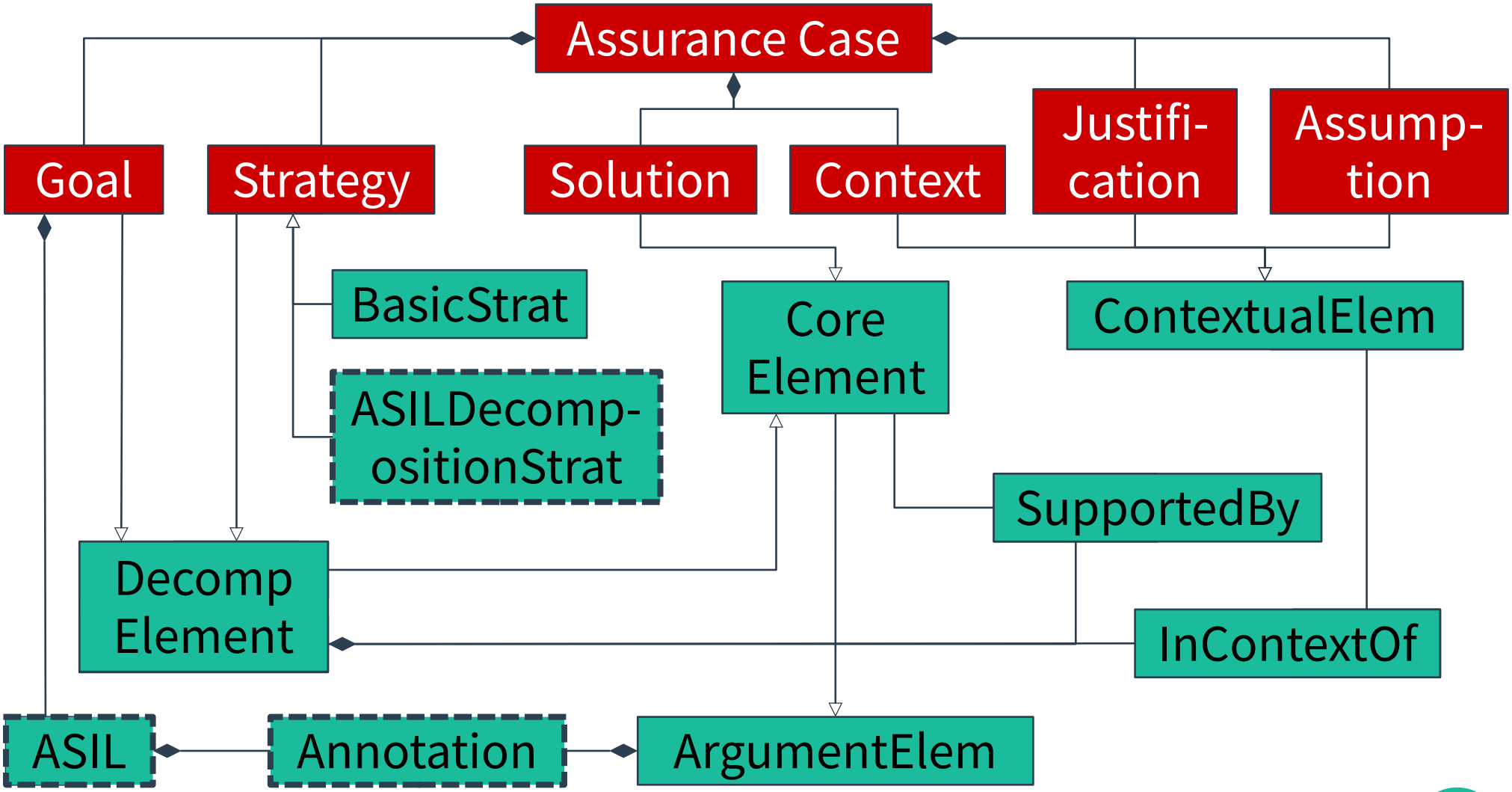
Extensions for MMINT-A

- Assurance Case Metamodel
- Assurance Case Editor
- Assurance Case Slicers
- Change Impact Assessment Workflow

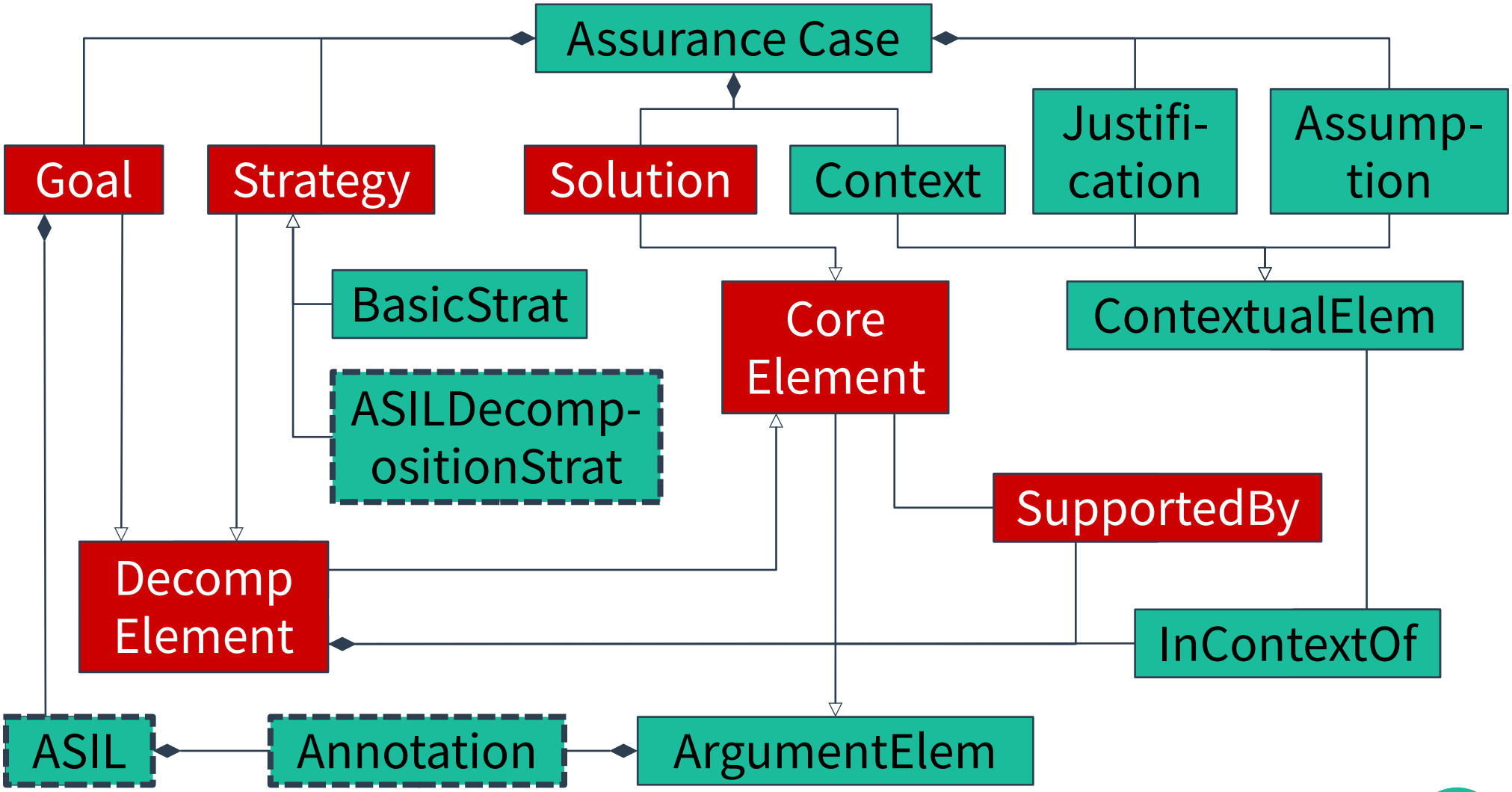
Assurance Case (GSN) Metamodel



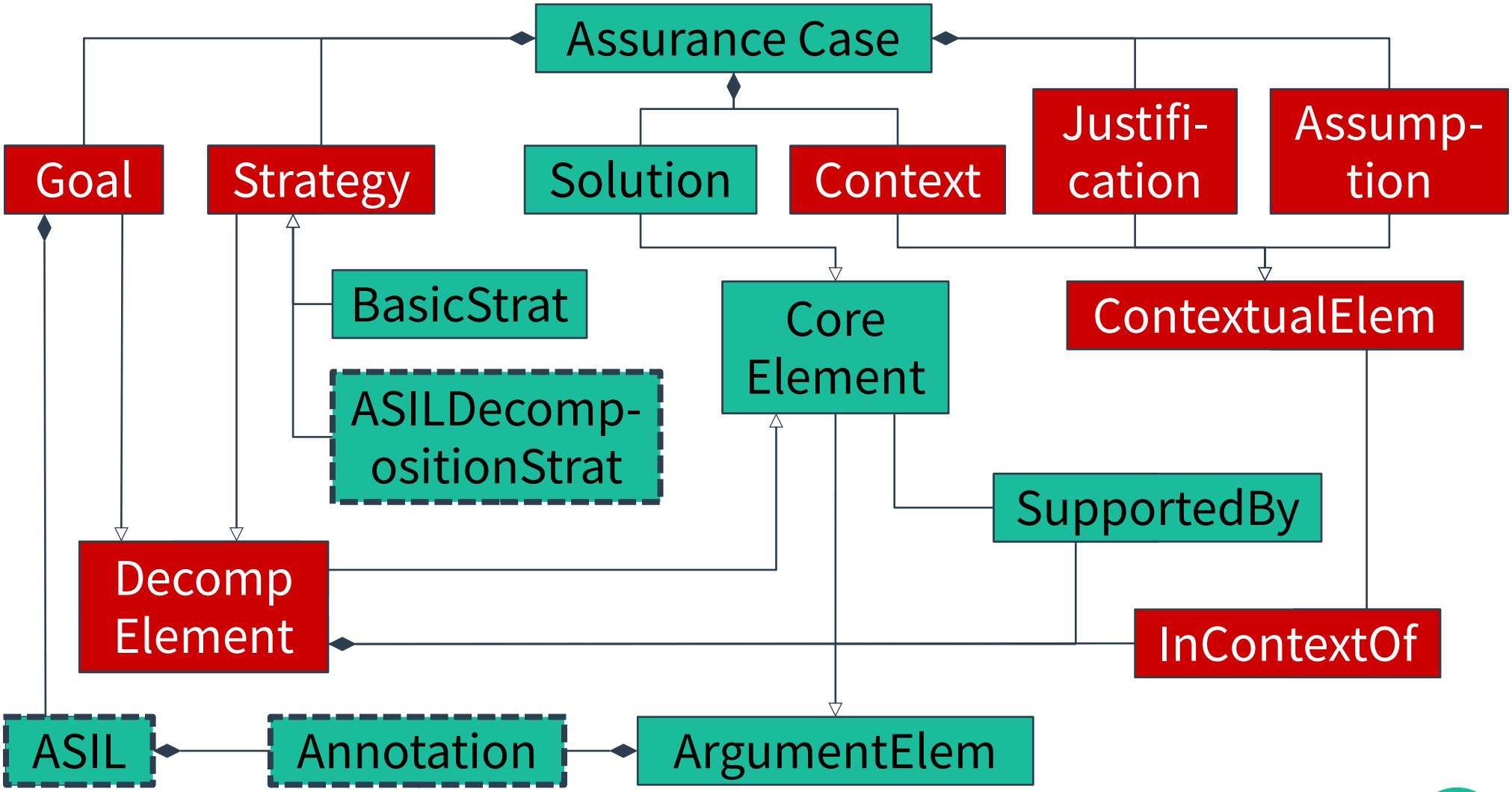
Assurance Case (GSN) Metamodel



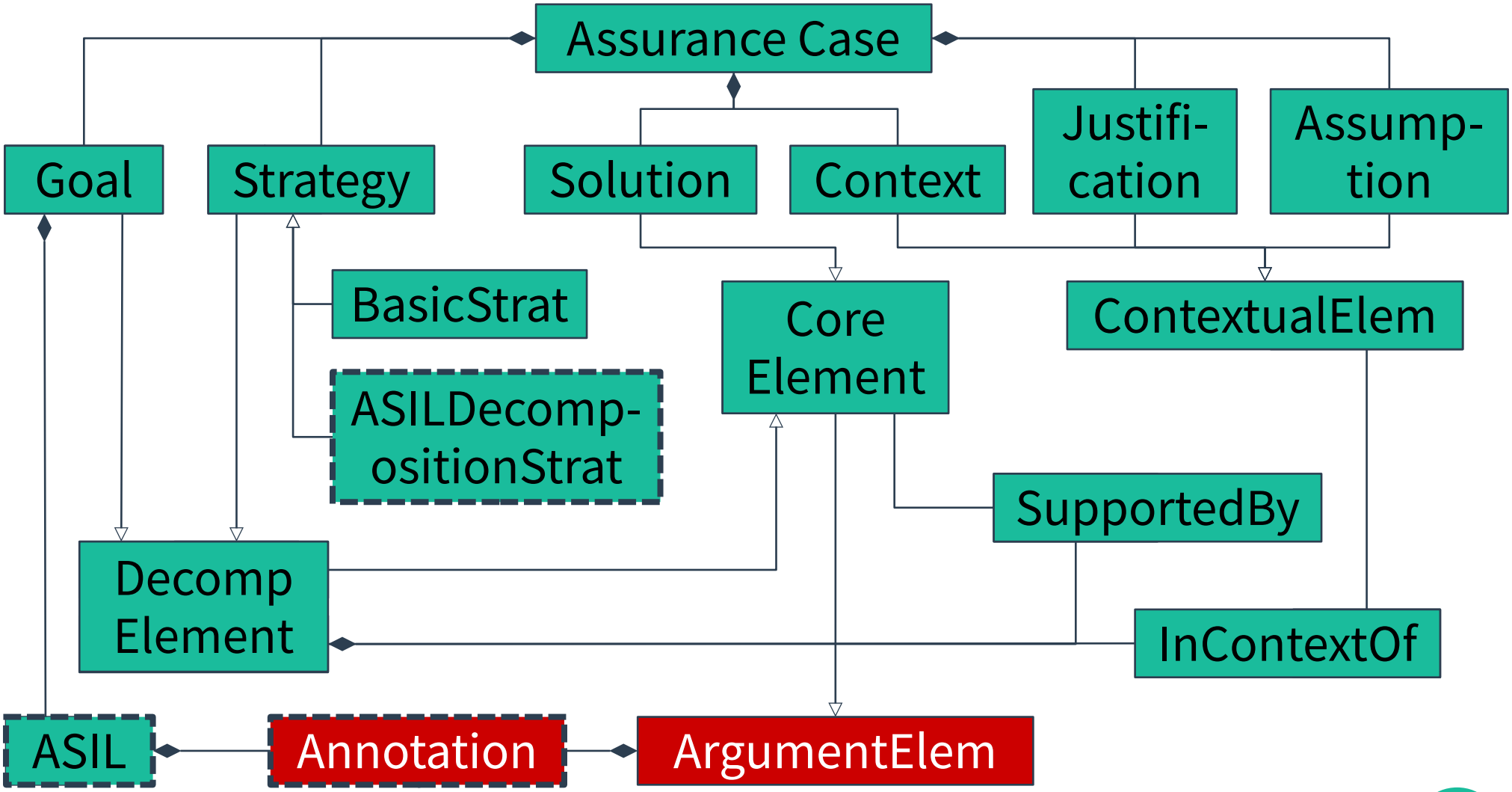
Assurance Case (GSN) Metamodel



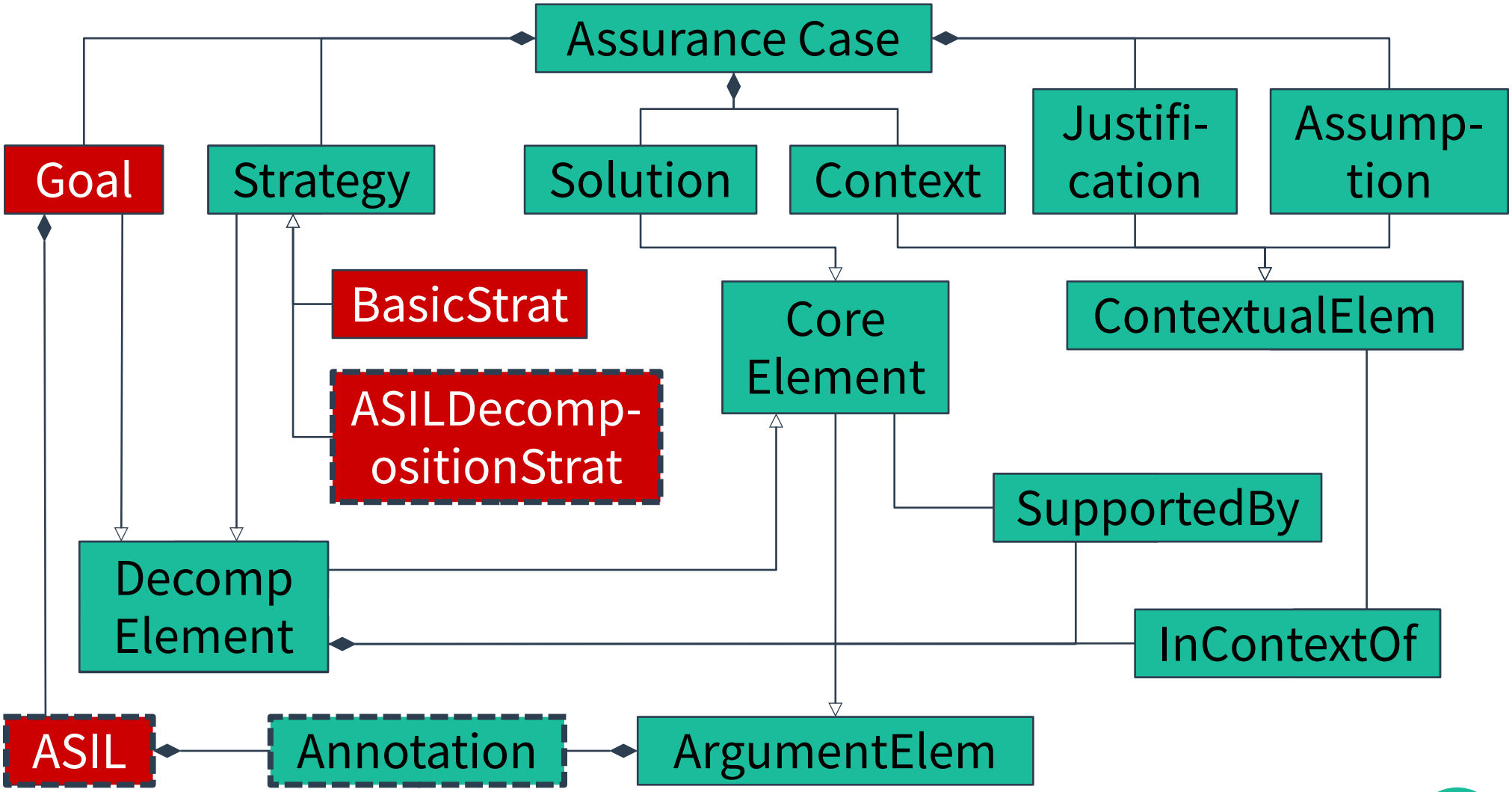
Assurance Case (GSN) Metamodel



Assurance Case (GSN) Metamodel



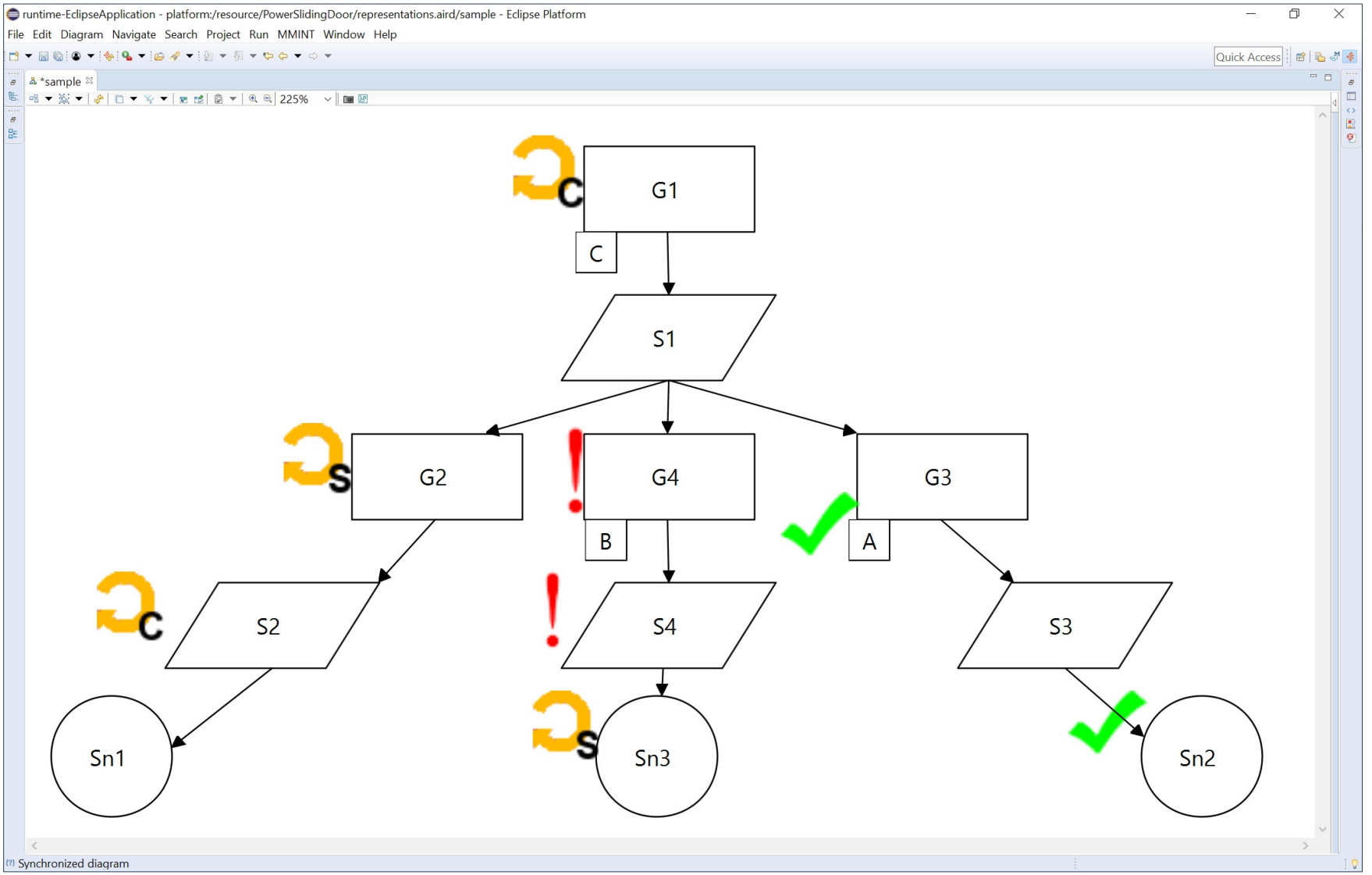
Assurance Case (GSN) Metamodel



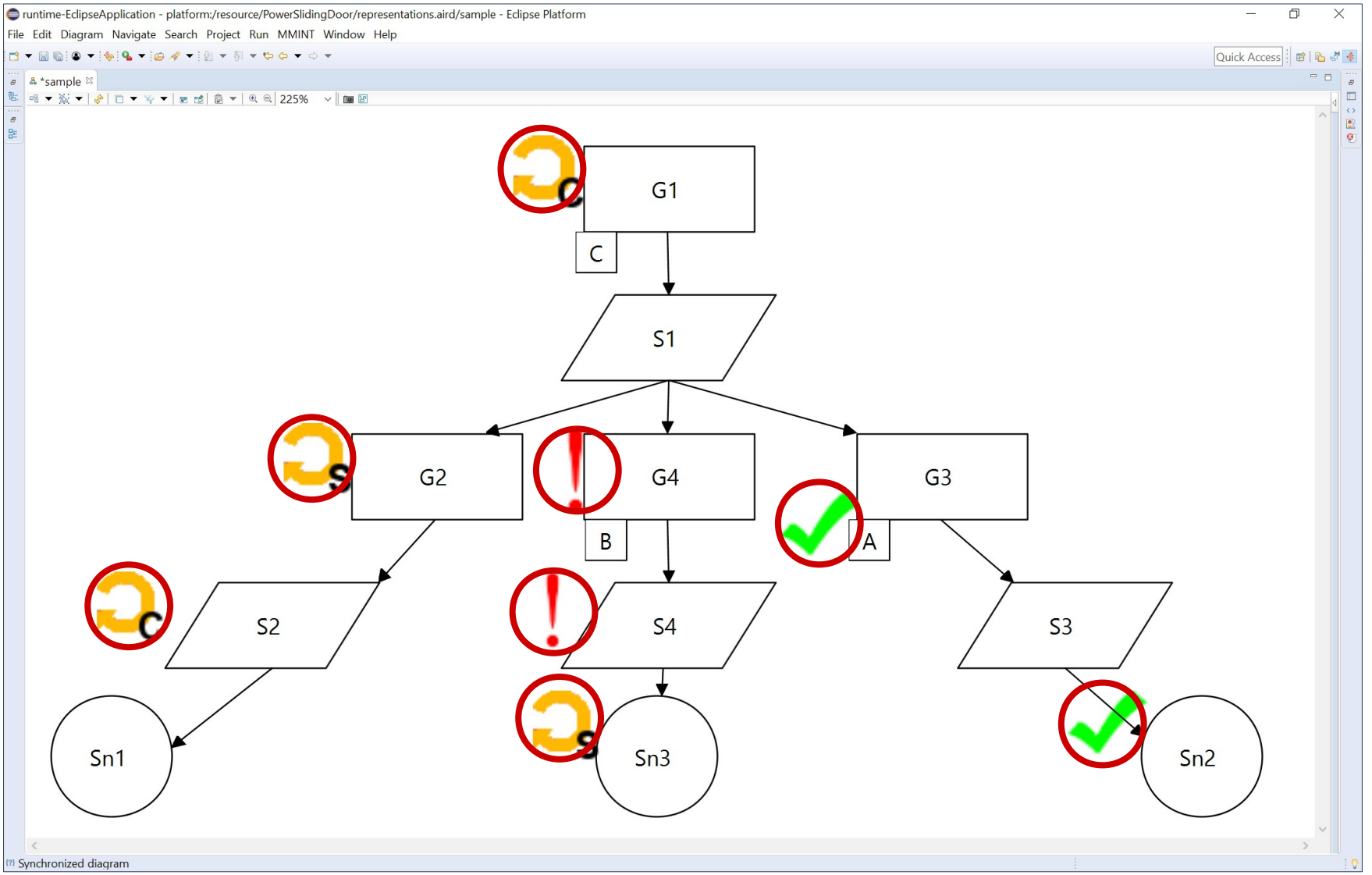
Extensions for MMINT-A

- Assurance Case Metamodel
- Assurance Case Editor
- Assurance Case Slicers
- Change Impact Assessment Workflow

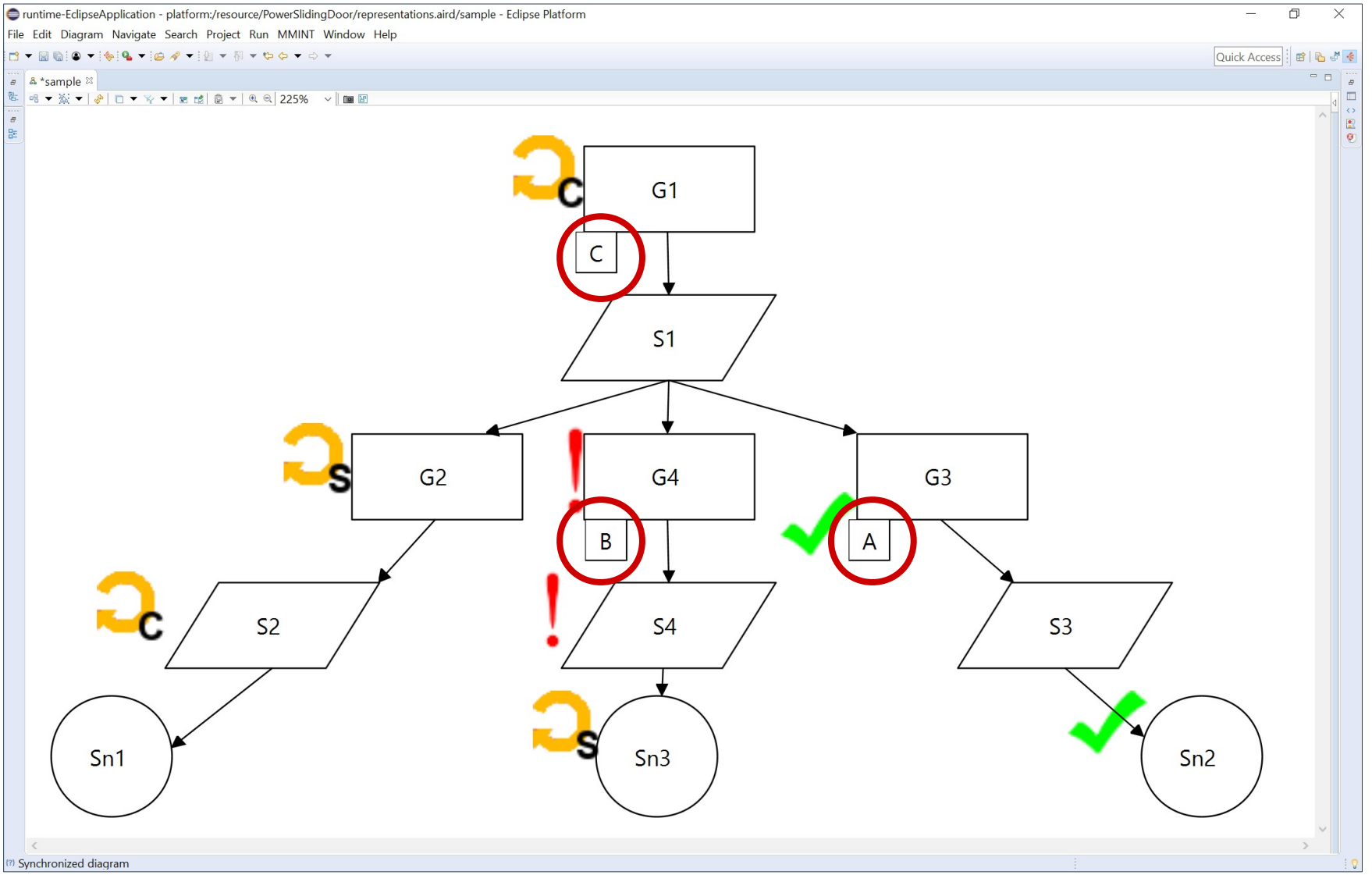
Assurance Case Editor



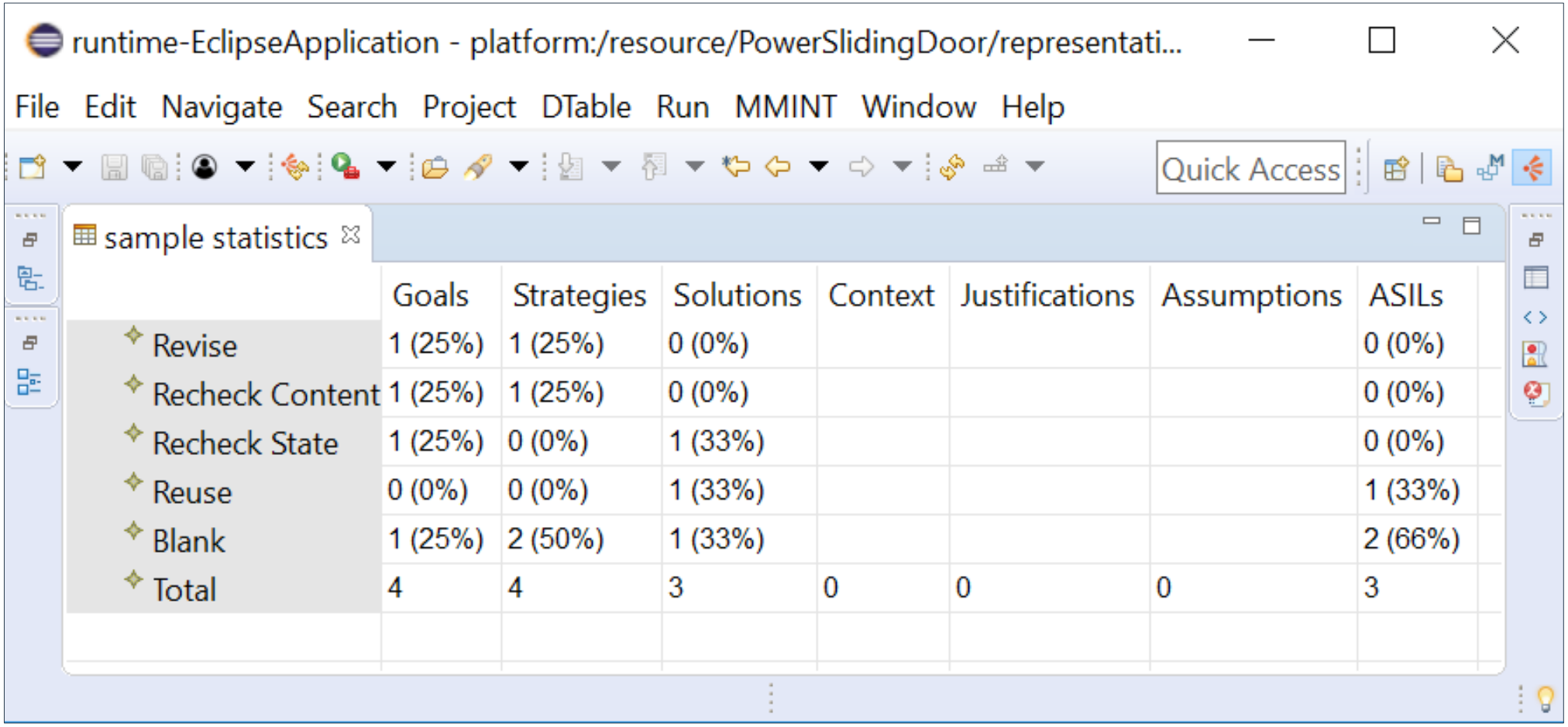
Assurance Case Editor



Assurance Case Editor

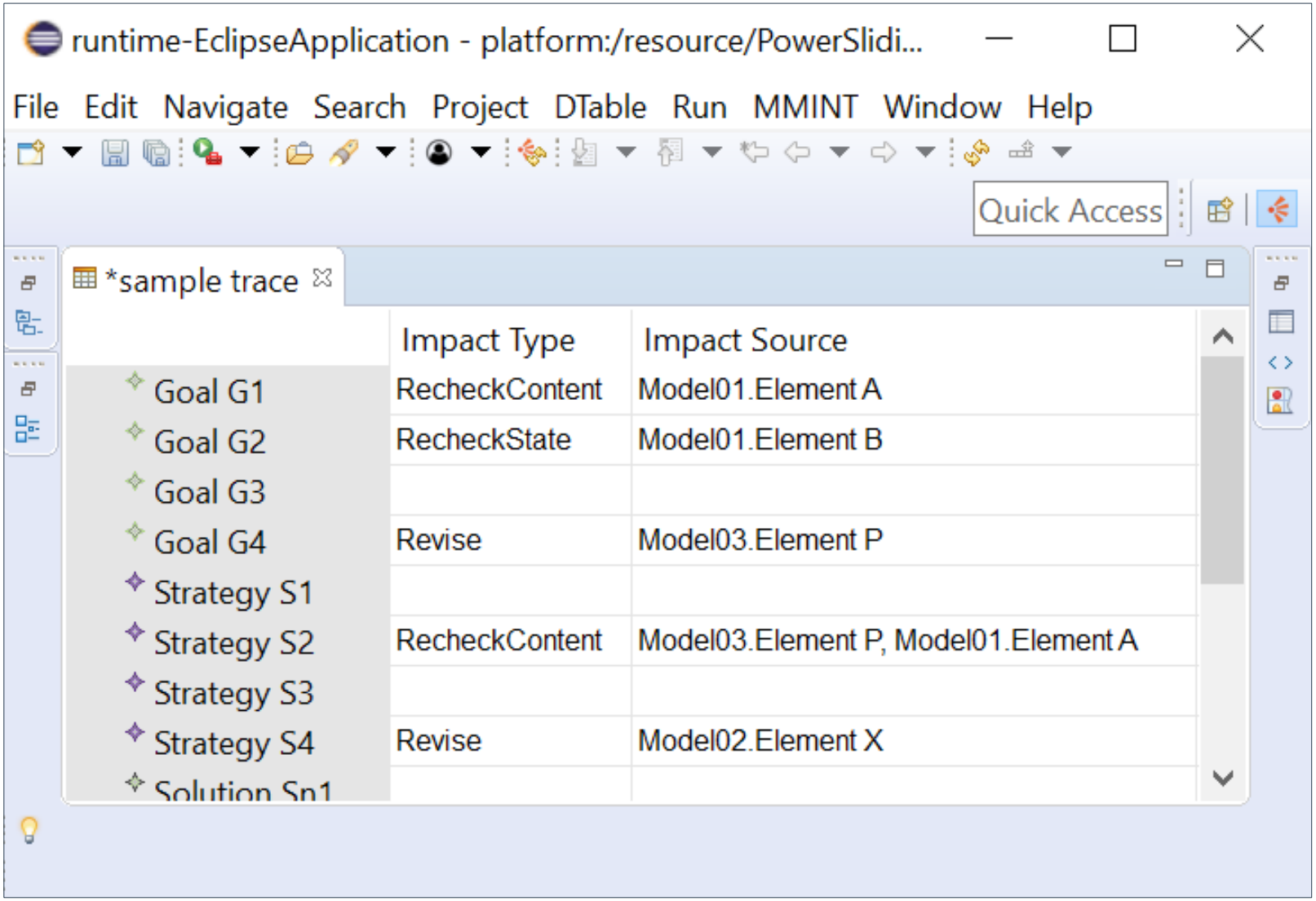


Impact Summary Table



	Goals	Strategies	Solutions	Context	Justifications	Assumptions	ASILs
Revise	1 (25%)	1 (25%)	0 (0%)				0 (0%)
Recheck Content	1 (25%)	1 (25%)	0 (0%)				0 (0%)
Recheck State	1 (25%)	0 (0%)	1 (33%)				0 (0%)
Reuse	0 (0%)	0 (0%)	1 (33%)				1 (33%)
Blank	1 (25%)	2 (50%)	1 (33%)				2 (66%)
Total	4	4	3	0	0	0	3

Impact Trace Table



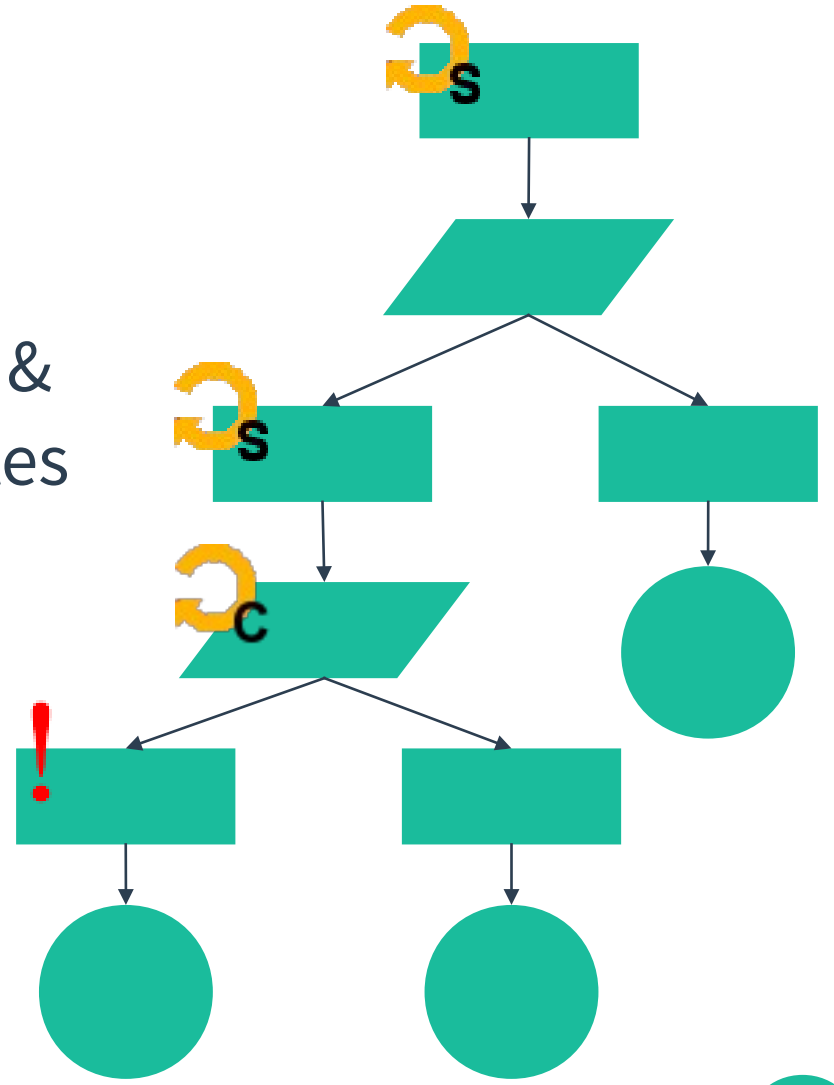
	Impact Type	Impact Source
Goal G1	RecheckContent	Model01.Element A
Goal G2	RecheckState	Model01.Element B
Goal G3		
Goal G4	Revise	Model03.Element P
Strategy S1		
Strategy S2	RecheckContent	Model03.Element P, Model01.Element A
Strategy S3		
Strategy S4	Revise	Model02.Element X
Solution Sn1		

Extensions for MMINT-A

- Assurance Case Metamodel
- Assurance Case Editor
- Assurance Case Slicers
- Change Impact Assessment Workflow

Assurance Case Slicers

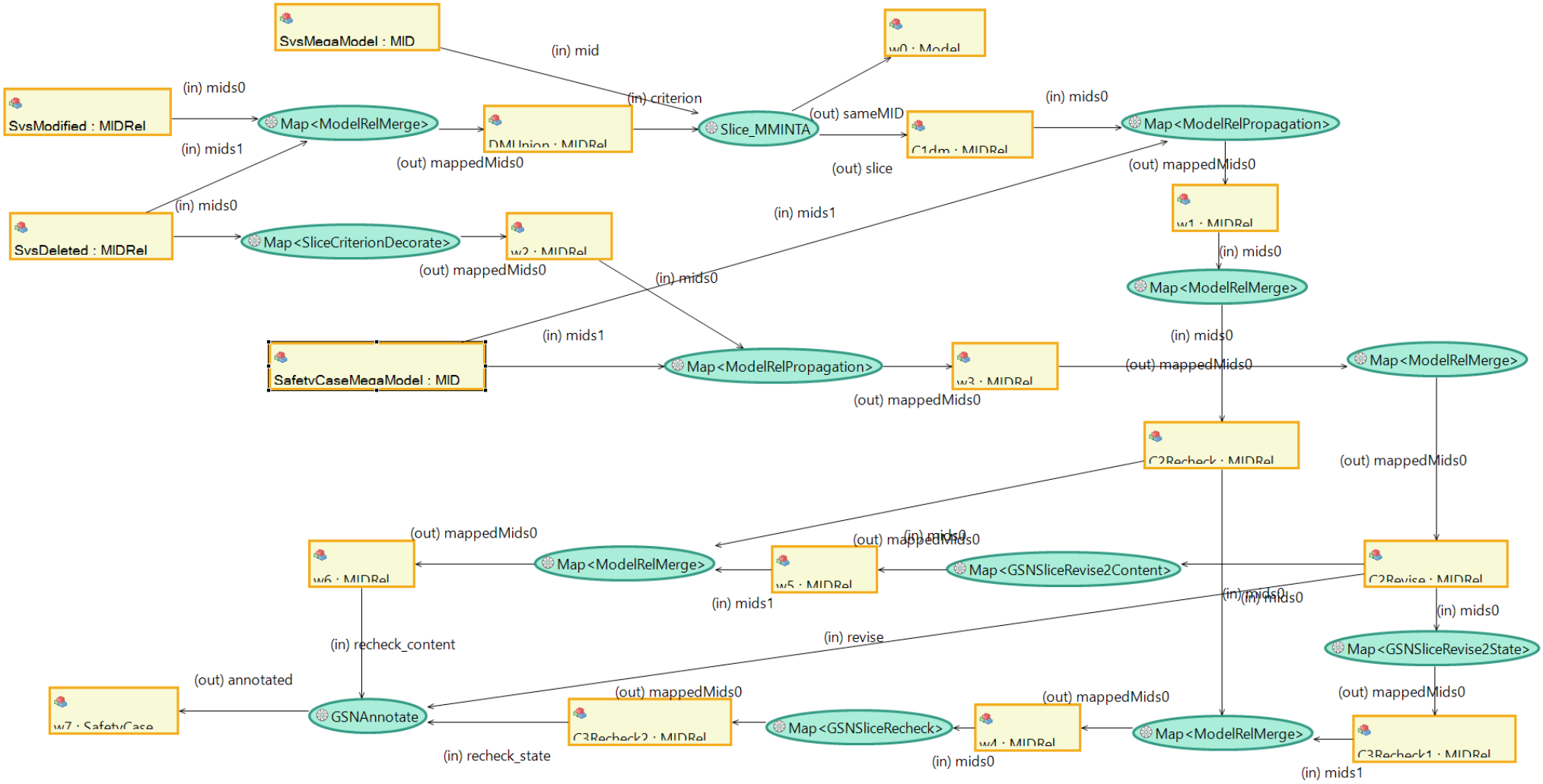
- **Propagates change impact**
- **Based on If-Then rules, e.g.:**
  - Revise goal →
    Recheck strategy contents &
    Recheck ancestor goal states



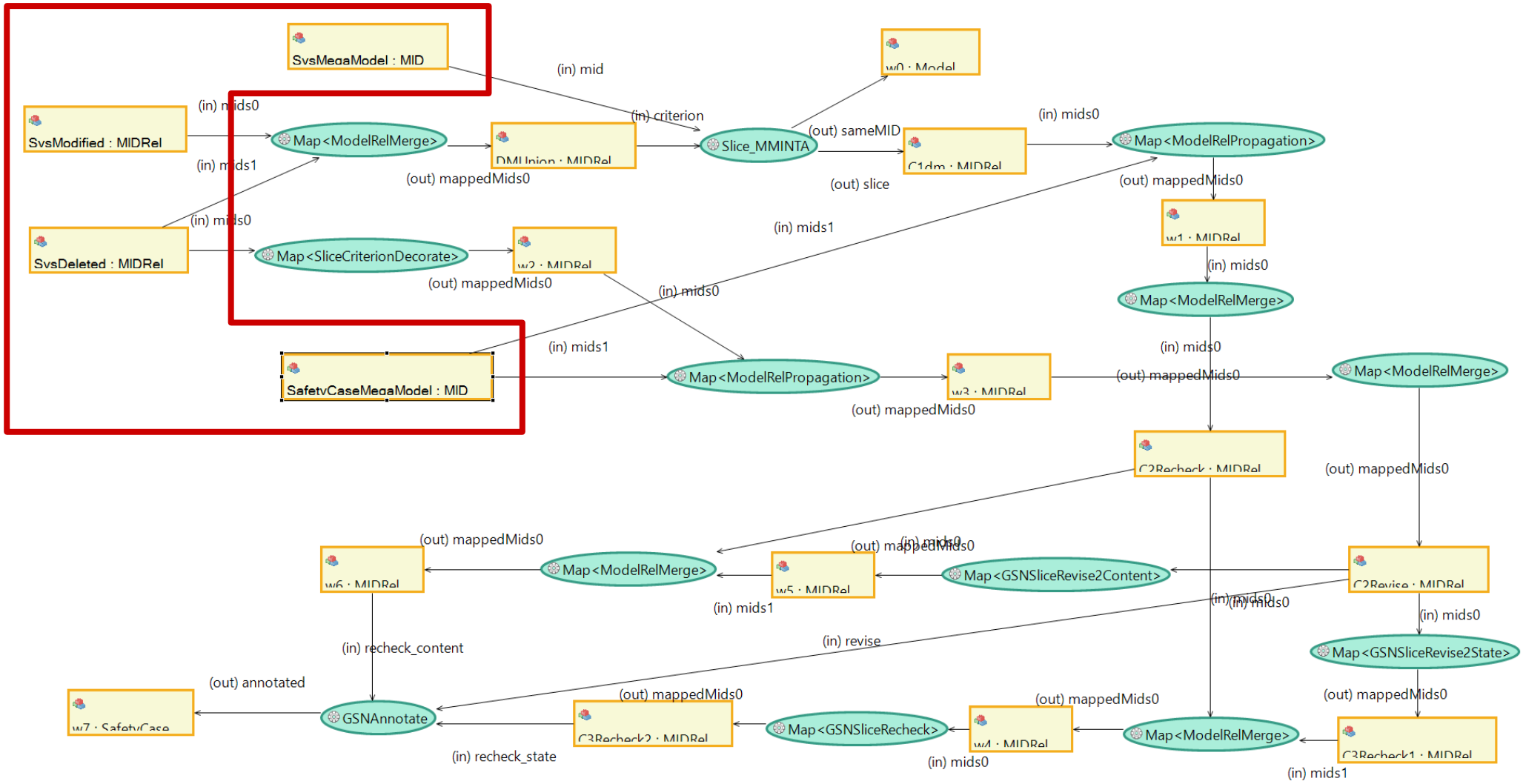
Extensions for MMINT-A

- Assurance Case Metamodel
- Assurance Case Editor
- Assurance Case Slicers
- Change Impact Assessment Workflow

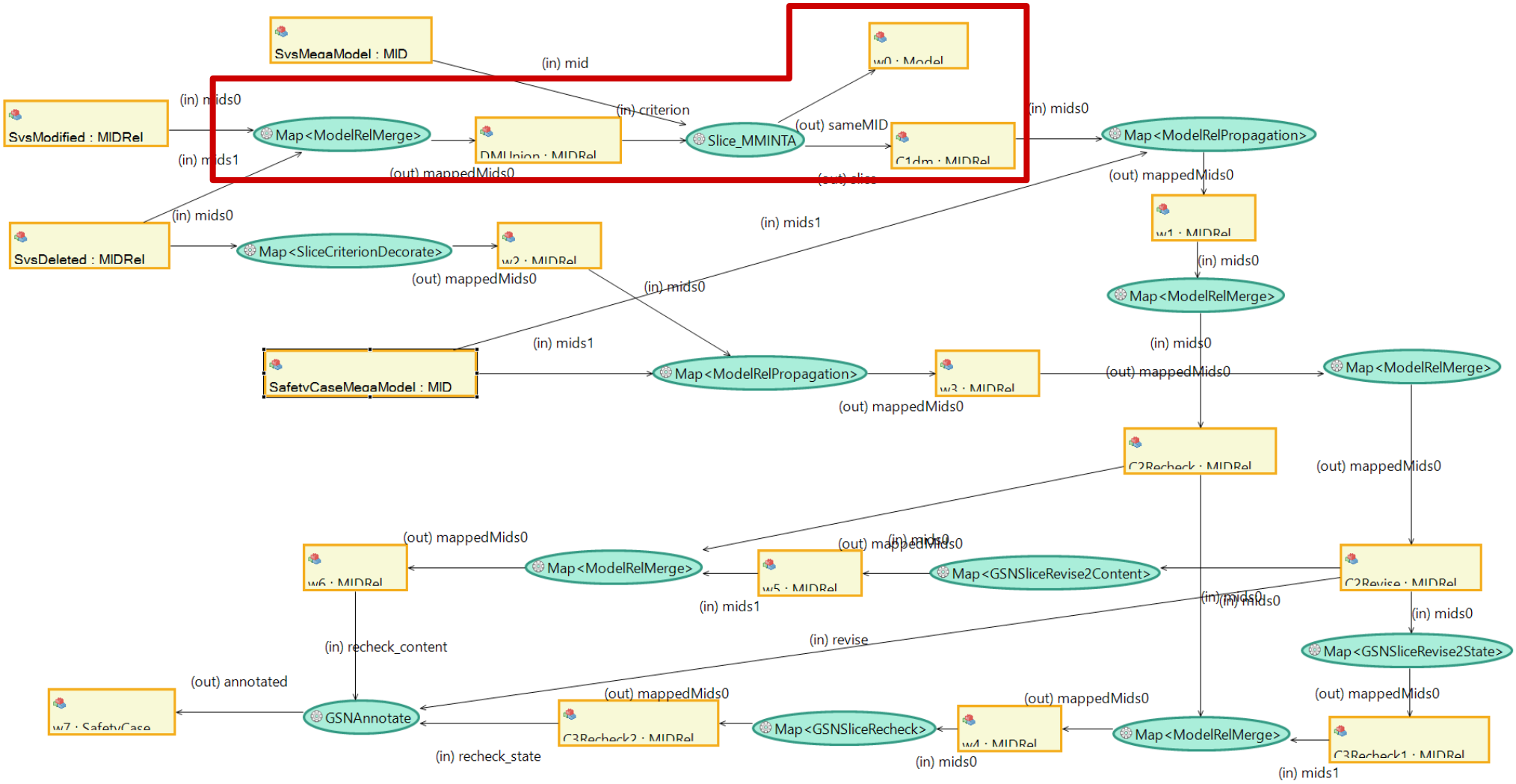
Change Impact Assessment Workflow



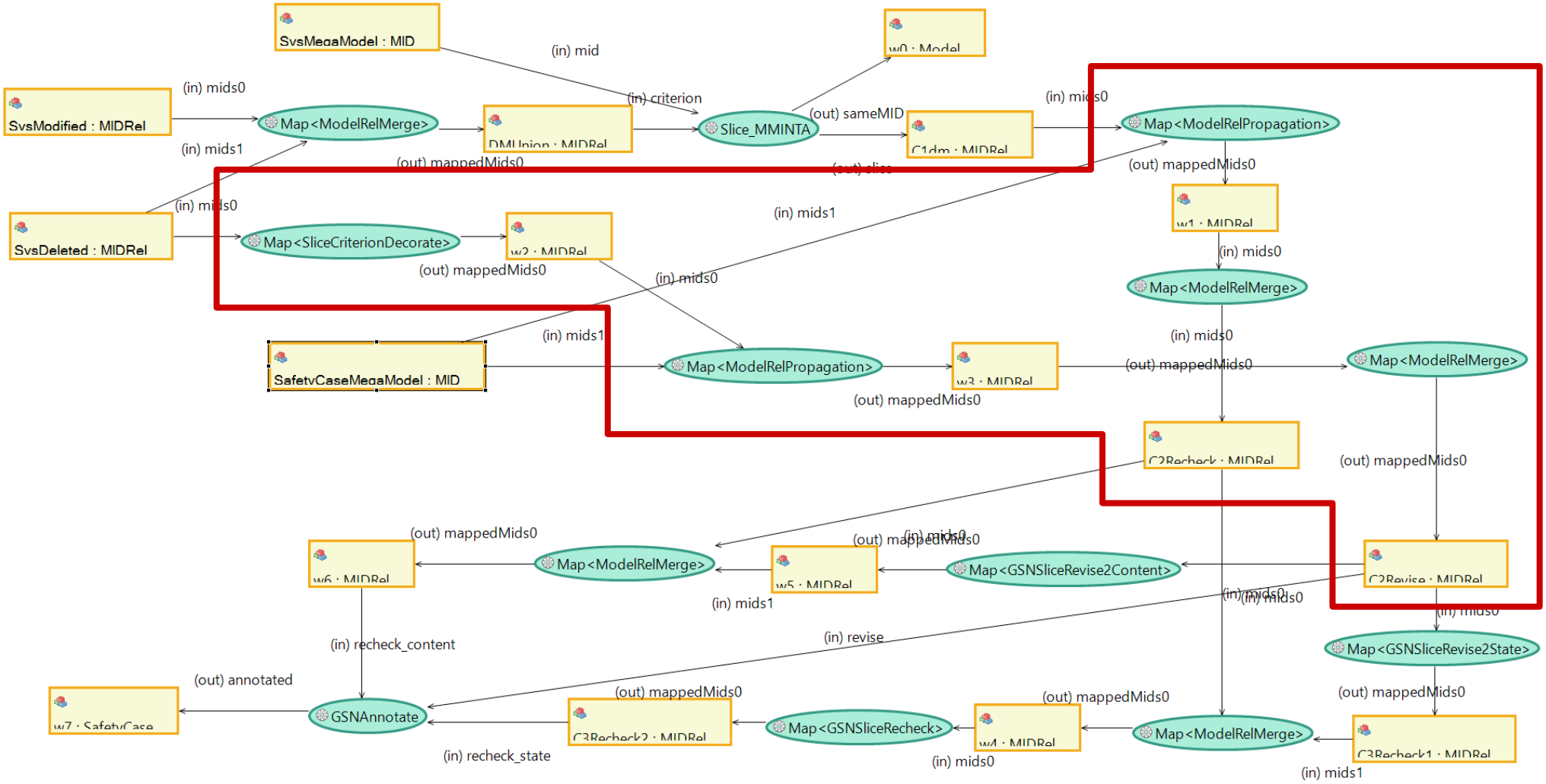
Change Impact Assessment Workflow



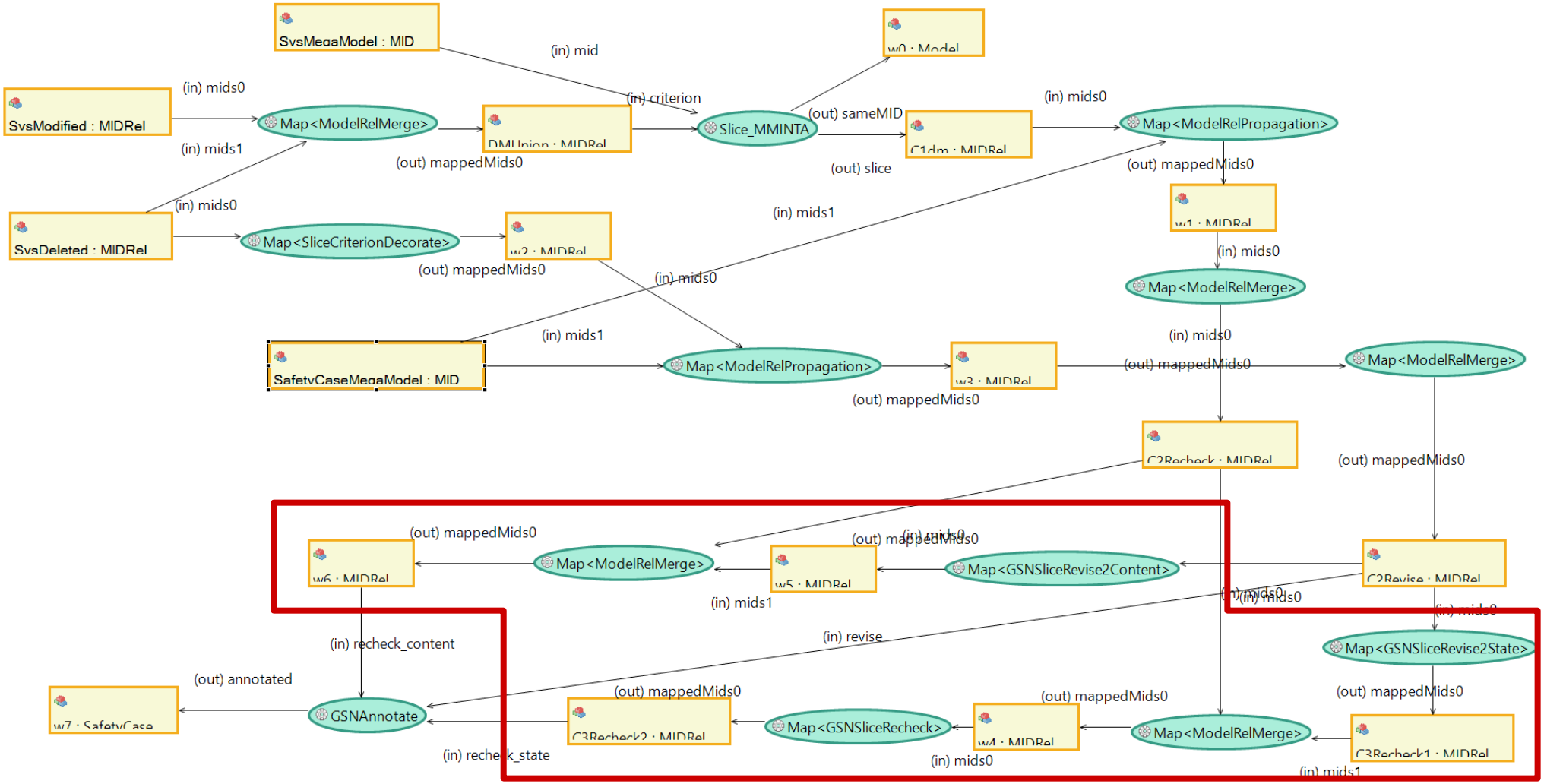
Change Impact Assessment Workflow



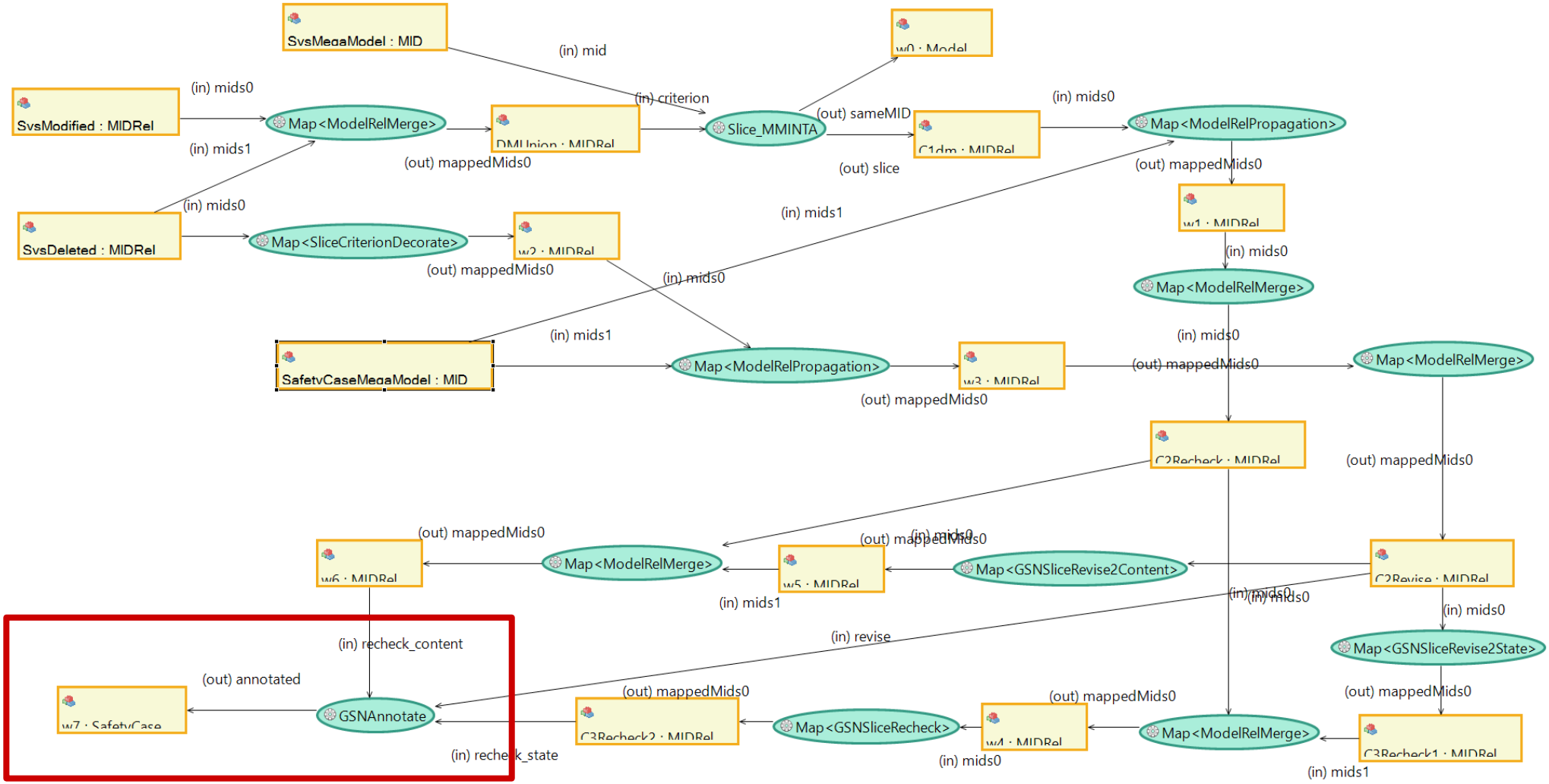
Change Impact Assessment Workflow



Change Impact Assessment Workflow



Change Impact Assessment Workflow

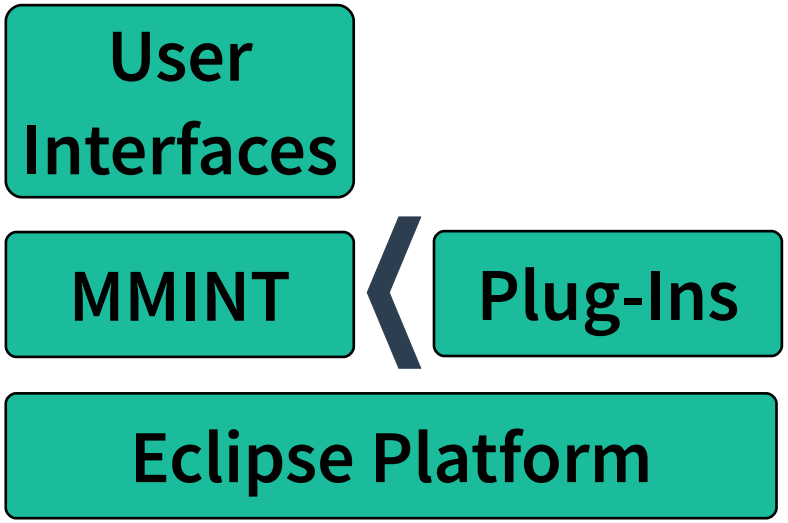


Case Study: Power Sliding Door

- **ISO 26262 Example**
- **Safety Requirement**
  - Door remains closed if travelling above 15 km/hr

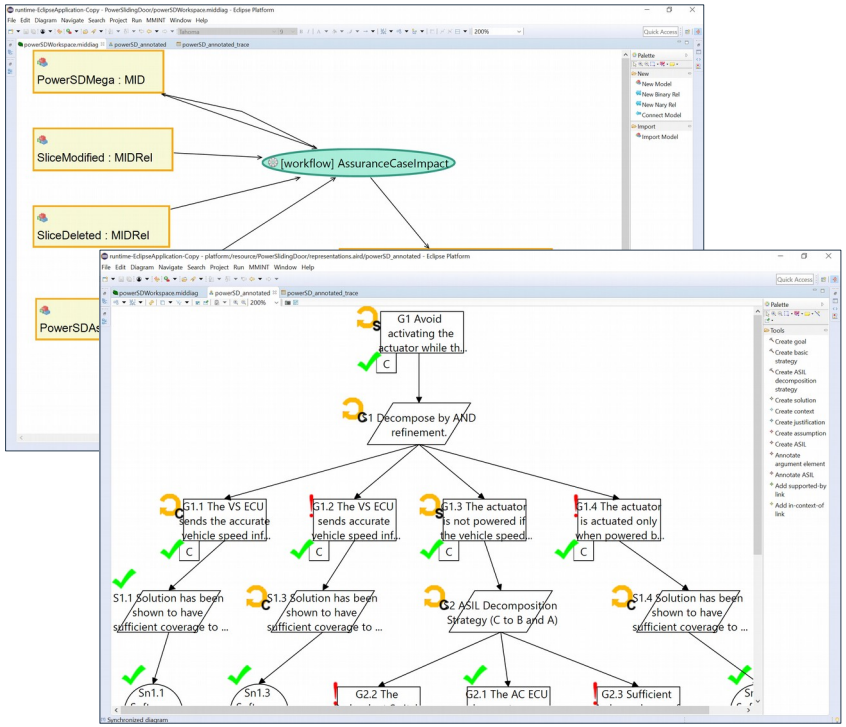
Summary

- **MMINT**
  - Model Management Interactive
- **Extensions for MMINT-A**
  - Metamodel
  - Editor
  - Slicers
  - Change Impact Workflow
- **Future Work**
  - Support for SACM
  - Support for more model types
  - Validation of results & tool



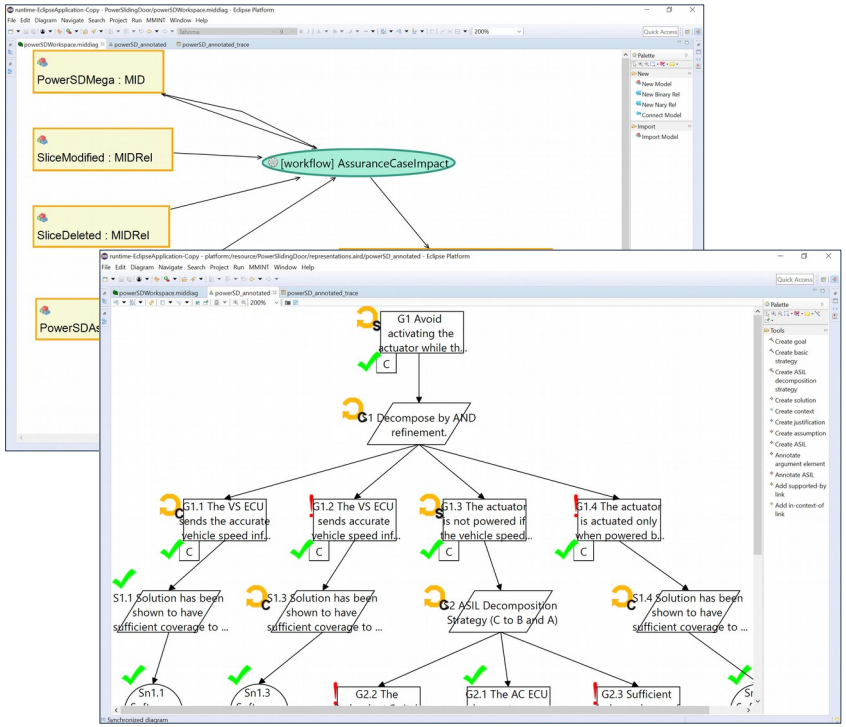
Summary

- **MMINT**
  - Model Management Interactive
- **Extensions for MMINT-A**
  - Metamodel
  - Editor
  - Slicers
  - Change Impact Workflow
- **Future Work**
  - Support for SACM
  - Support for more model types
  - Validation of results & tool



Summary

- **MMINT**
  - Model Management Interactive
- **Extensions for MMINT-A**
  - Metamodel
  - Editor
  - Slicers
  - Change Impact Workflow
- **Future Work**
  - Support for SACM
  - Support for more model types
  - Validation of results & tool



Questions?

MMINT-A: A Tool for Automated Change Impact Assessment on Assurance Cases

Nick Fung, Sahar Kokaly, Alessio Di Sandro,
Rick Salay, Marsha Chechik

Repo: https://github.com/nlsfung/MMINT
Manual: https://github.com/nlsfung/MMINT/wiki

UNIVERSITY OF TORONTO

McMaster University