SME Model Lifecycle Managment

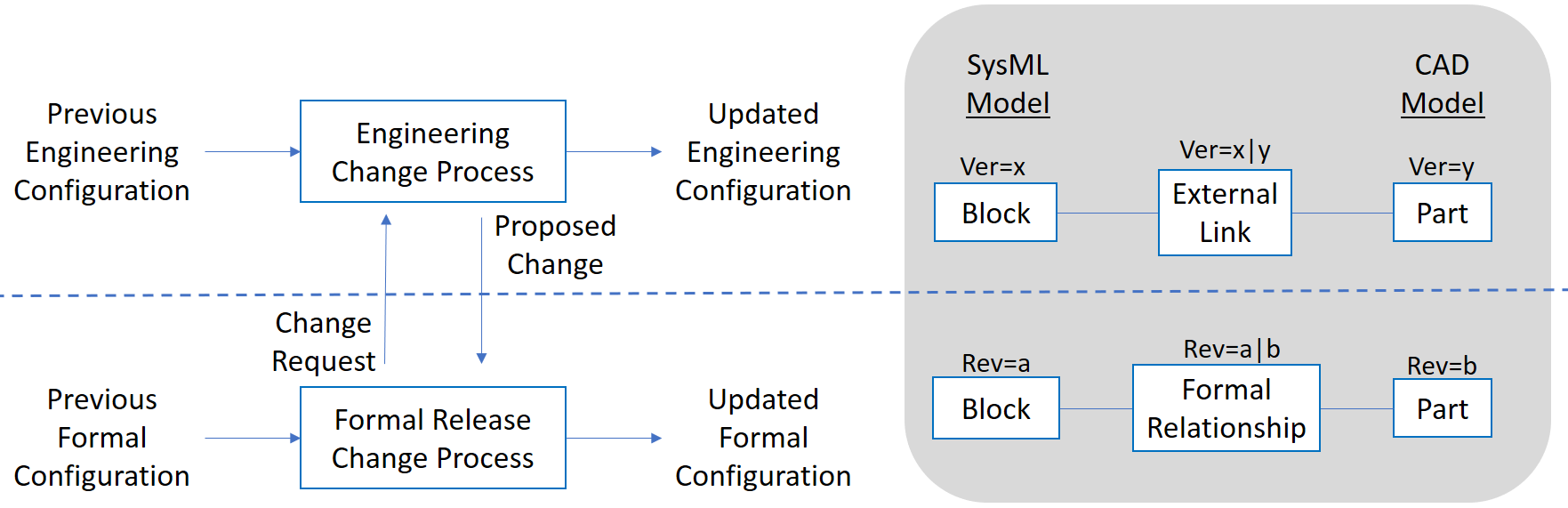
# SCOPE

Model Lifecycle Management (MLM) manages the Engineering Change Process to provide Proposed Changes in response to Engineering Change Requests from the Formal Release Change Process. It is a governance process synchronizing the create, read, update, and delete (CRUD) operations on heterogeneous models within the supporting modeling tools and model repositories, throughout the system development lifecycle. This is accomplished through the management of Model Configuration Items (MCIs) including versions, variations, and configurations of models, simulations, analysis results, and the tools that are used by multiple geographically dispersed users. In addition, MLM includes the management of all the metadata associated with the models, tools, and analysis results including who made the change, what changes were made, when and why, as well as information regarding the application of the model.

## KEY CONCEPTS

**Fig 1. Integrated System Model (ISM) Lifecycle Management**

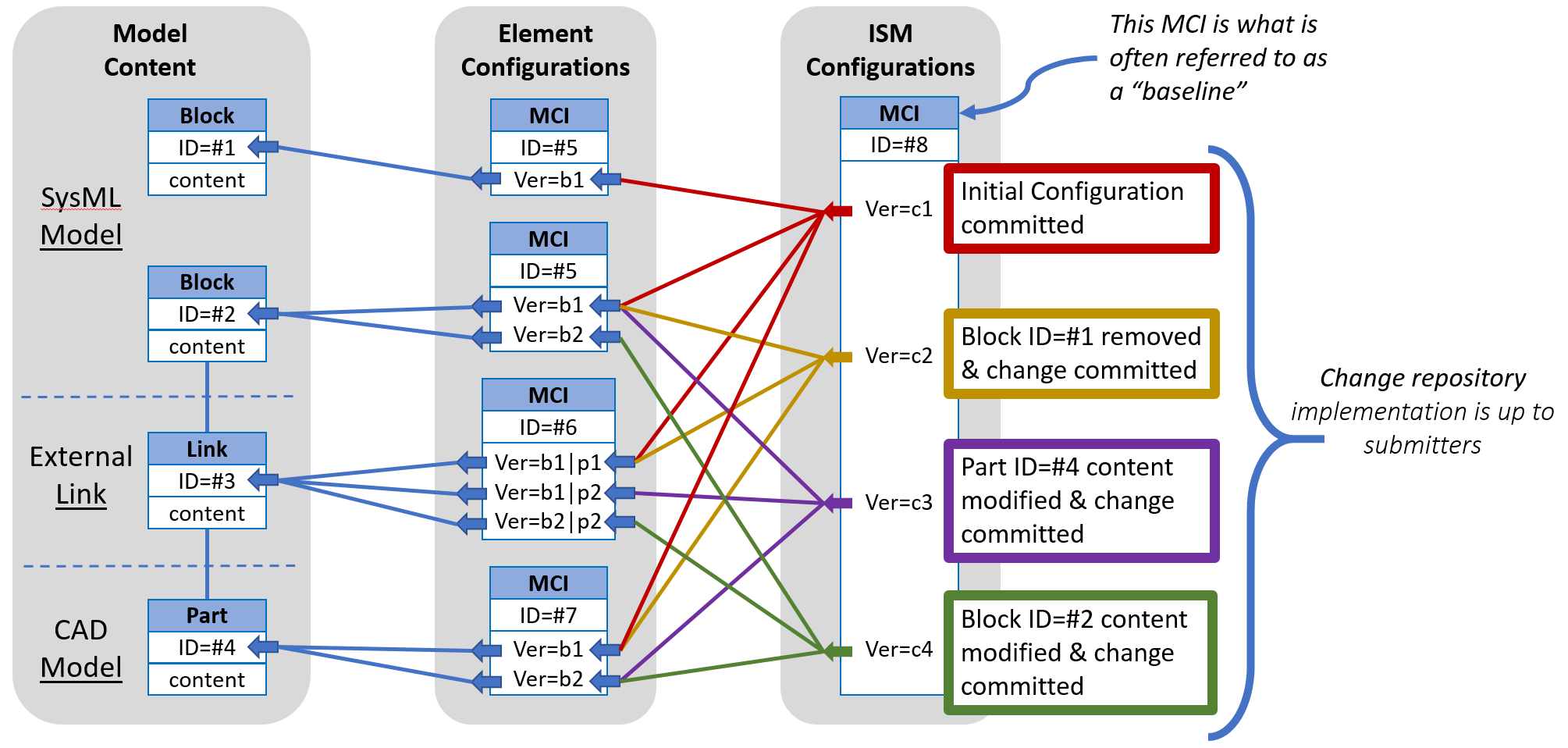
**Fig 2. Engineering Change vs. Formal Release Change**



Additional details:

1. Engineering Change Process is Iterative and can support various design methodologies (e.g. Agile)
2. Both processes can exist in the same or in two different environments with different update frequencies. - implementation is up to submitters

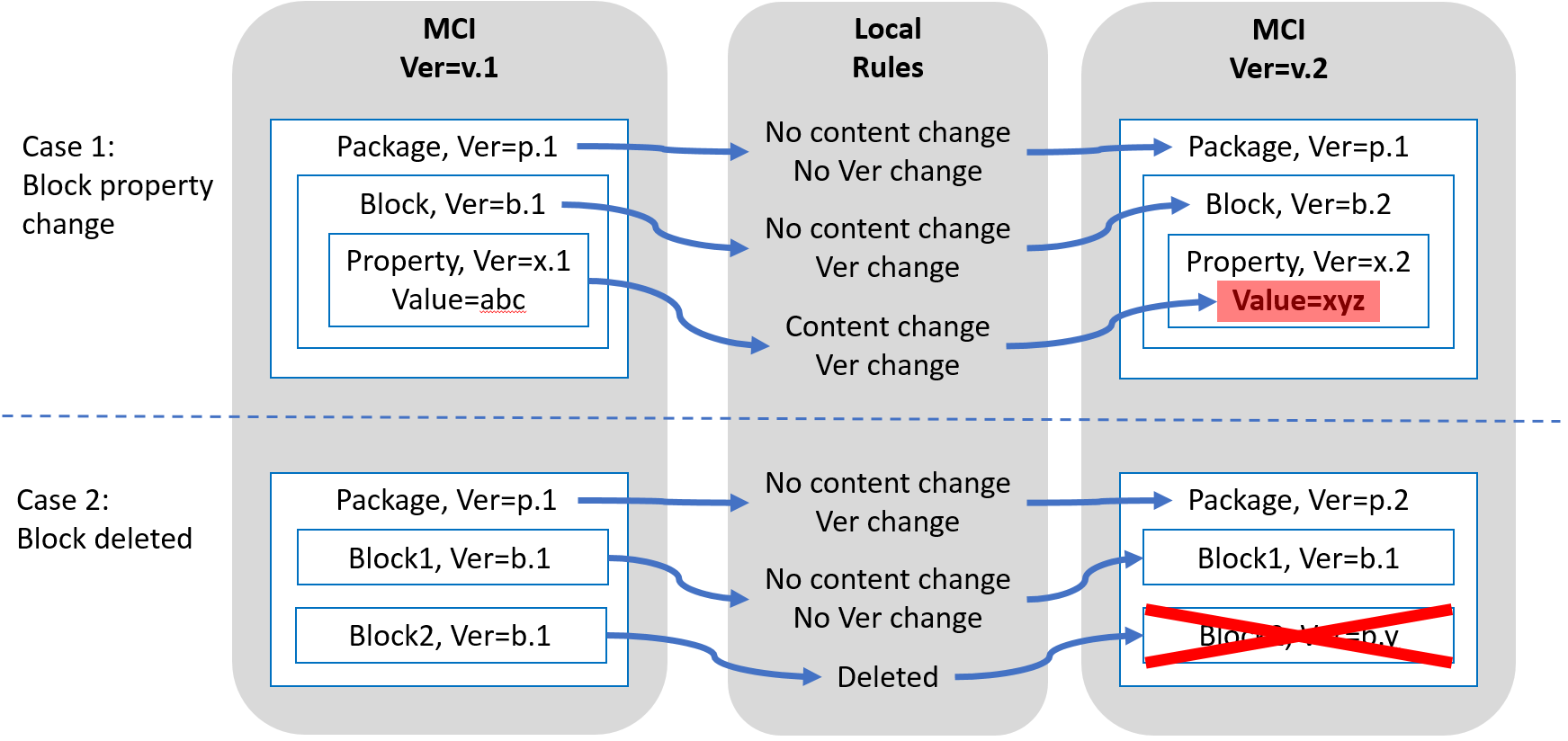
**Fig 3. Model Configuration and Model Configuration Items (MCI)**



MCI implementation is up to submitters including:

1. Not implemented in SysML, versions managed completely outside of SysML model
2. MCIs are implemented in SysML (metadata extensions) but only as versions of model elements – with configurations (higher level MCIs) managed completely outside of SysML model
3. MCIs are implemented is SysML (metadata extensions) as versions of model elements and as configurations (higher level MCIs)

**Fig 4. Example of SysML Package and MCI versioning**



Additional details:

1. Versioning rules of MCI in a hierarchy depend on the element type referred to by that MCI and on the local rules
2. Changes to MCI content elements results in MCI version increment on change commit.

# Requirments

The next-generation modeling language must be capable of being managed in a heterogeneous and distributed modeling environment. The ability to manage change to the model, where multiple users are collaborating on a single model, and when some elements of the model are defined outside of the modeling tool (ex: requirements or physical parts), is challenging enough. This requires basic model change management that includes effective means for evaluating and integrating changes from multiple users and sources, while maintaining a history of all changes to the MCIs. These challenges are magnified when multiple models and tools are all part of the collaboration. The ability to integrate with Product Lifecycle Management (PLM) environments, which enable versioning, configuration, and variant management of the model structures that are linked to the other domain structures (ex: MCAD, ECAD, or ALM), is also a fundamental SME requirement.

<http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:sysml_assessment_and_roadmap_working_group>

## METADATA

1. ID – a unique global identifier of any element
2. MCI – a new element
   1. Has an ID
   2. Has a Version
      1. Changes to a MCI contained element will result in a MCI version increment on change commit action. (e.g. if a property of a block within a MCI changes, then the MCI for that block and the block’s properties version are incremented.)
      2. In the absence of Configuration Management, the default version is the latest version.
   3. Refers to a one or more elements including other MCI

## Service Requirements

1. Manage changes at the MCI level
   1. Define MCI content (one or more)
   2. Authorize MCI update including version and elements (ex: ECO)
   3. Manage user Authorizations
      1. Includes individual permissions (ex: “can edit”)
      2. Works in conjunction with Data Protection services
   4. Define Version
   5. Log changes
   6. Compare and identify differences
2. Manage baseline/branch/merge/rebase (independent of Version)
   * 1. Create baseline
     2. Create branch of a baseline
     3. Update branch of a baseline
     4. Merge branch to trunk of a baseline
     5. Rebase, apply trunk to branch
     6. ~~De-conflict merge resources~~
3. Manage Data Protection Controls (e.g. access permissions, roles, data rights, markings)
   1. Create data protection controls
   2. Read data protection controls
   3. Update data protection controls
   4. Delete data protection controls (e.g. access permissions, roles, data rights, markings)
4. Extend metadata terms (Cross cutting requirement to add meta data across all SysML 2.0 working groups.)
   1. Create new metadata term (e.g. term, type, owner)
   2. Exchange new metadata terms; Model exchange should include mechanism for preserving user defined metadata within the context of model exchange.
   3. Metadata shall be persistent either in the data model of authoring tool or in a separate repository owned by the service provider (applies to services that authoring tool utilizes).

# Glossary

1. **Total System Model** (TSM) – A set of linked models (e.g. architecture, behavior, simulation, software, CAD, CAE, etc.) that together define a complete system.
2. **Integrated System Model** (ISM) – A composition of the System Architecture Model and both internal and external reference links. External reference links can link to MCIs within the same or other engineering domain models.
3. **System Model Management** (SMM) – Model Management Services for the Integrated System Model (ISM) which includes the TSM and the links between the TSM and other specialty models. It does not replace the linked model’s native configuration management tool.
4. **Model Lifecycle Management** (MLM) - Manages the Engineering Change Process to provide Proposed Changes in response to Engineering Change Requests from the Formal Release Change Process.
5. **Model Configuration Item** (MCI) – A specific portion of the model (content and granularity) that is maintained in a controlled fashion, i.e. has a trackable ID and version history. MCI can be defined in different granularities, from an individual fine grained Model Element, a set of Model Elements, a set of Elements, to the entire Model. Any MCI can contain another MCI.
6. **Model Configuration** - an MCI version that designates versions of other MCIs, and links to versions of elements in external models (e.g., part, analysis component, …).
7. **Element** - Anything that has a unique ID including Model Elements, Links, Scripts, Constructs, Files, Data, …
8. **System Model** - A set of model elements contained in the SysML v2 model.
9. **Global Unique ID** (ID) – A unique identifier of an element. No standard is being imposed but one should be decided by the submitter.
10. **Version** - A state of an MCI at a given point in time as designated by the ***engineering change process***.
11. **Revision -** A state of an MCI at a given point in time as designated by the ***formal release change process***.

# Open Issues

1. Clarify the concept of ID, UID, GUID across all SysML working groups.
   1. Is it globally unique?
   2. How does it relate to a Part #?
2. Clarify relationship between Manage authorization and data classification
3. Does model variant management belong in SysML – but outside of MLM as such? (Uwe)
4. Compatible vs incompatible change; changed or unchanged interfaces? (Michael)

# References

<http://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:model_lifecycle_management_for_mbse_v4.pdf>

The glossary of the INCOSE Systems Engineering Body of Knowledge (SEBoK)