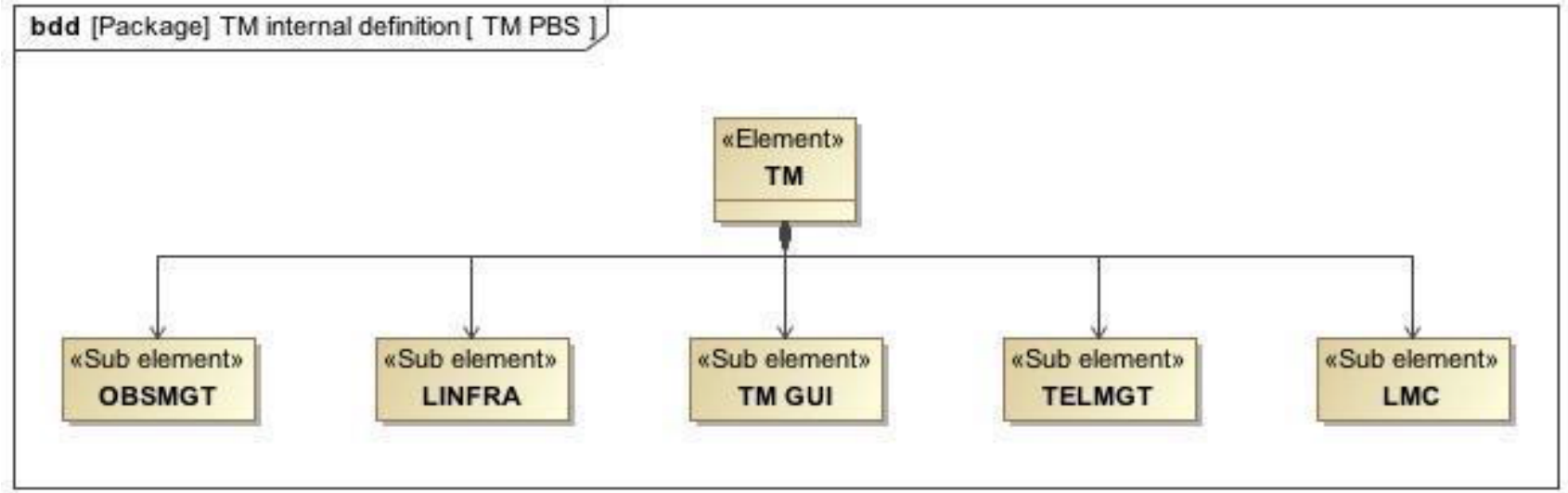
TM.LMC

M. Di Carlo, *M. Dolci*
and the TM.LMC team

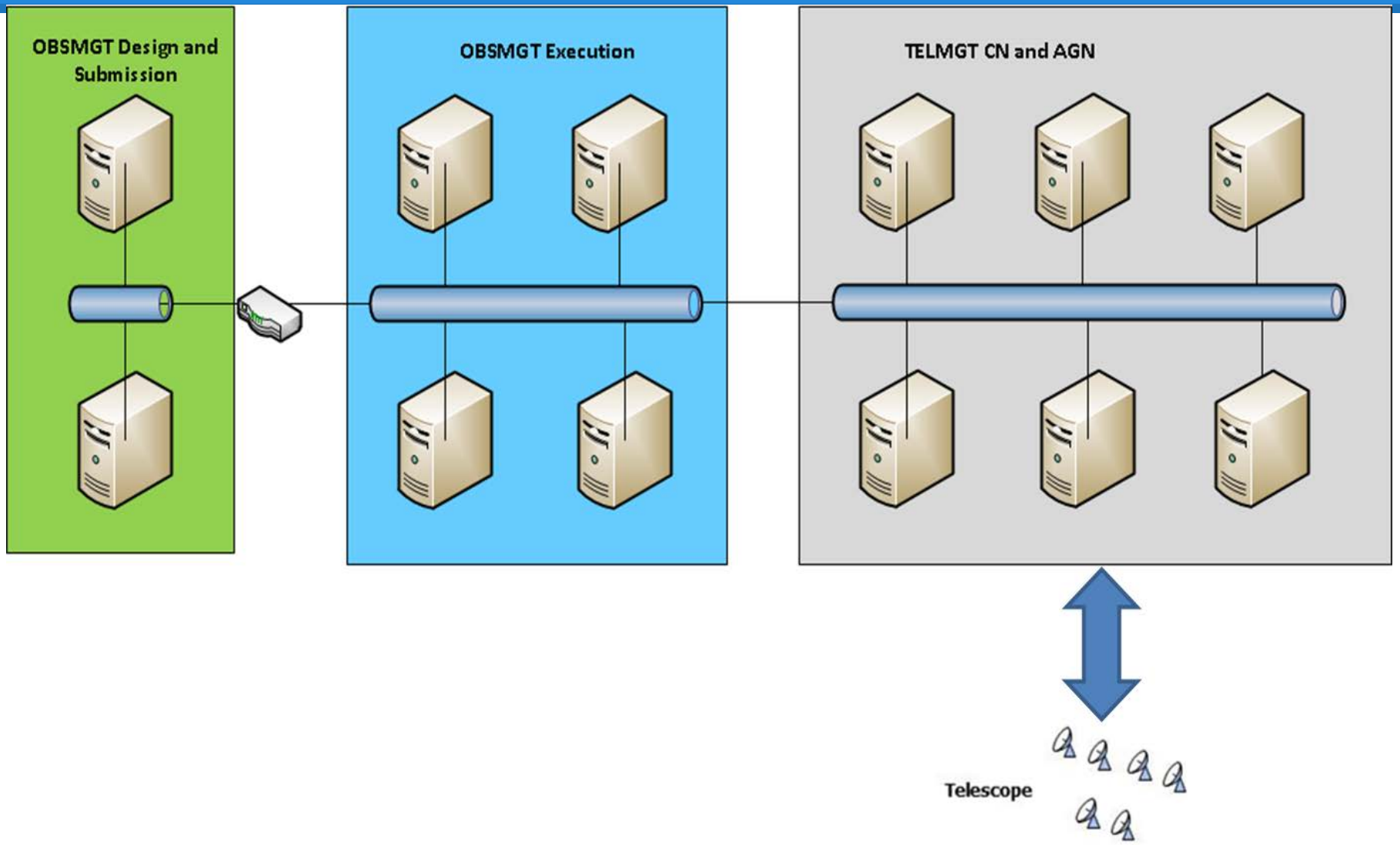
Outline

- TM architecture
- TM.LMC responsibilities
- TM.LMC functions
- TM.LMC Architectural Model
- Local M&C concept
- Updating strategy
- Load balancing, scalability, reliability and availability
- Abstract Data Model
- Common Framework Requirements

What is TM?


bdd [Package] TM internal definition [TM PBS]
«Element»
TM
«Sub element»
OBSMGT
«Sub element»
LINFRA
«Sub element»
TM GUI
«Sub element»
TELMGT
«Sub element»
LMC

Distributed software application



(TM.)LMC *general* responsibilities *(LSR)*

- Setup and configuration of sub-elements/components/LRUs
- Monitor the element as a system and its hardware and software components (rolled-up and drill-down views)
- Internal control (to achieve higher-level goals)
- Definition of higher-level control model (valid command sequences, their parameters and attributes)
- Implementation of a common state abstraction model
- Support for remote diagnostics
- Support for remote update/upgrade of the element

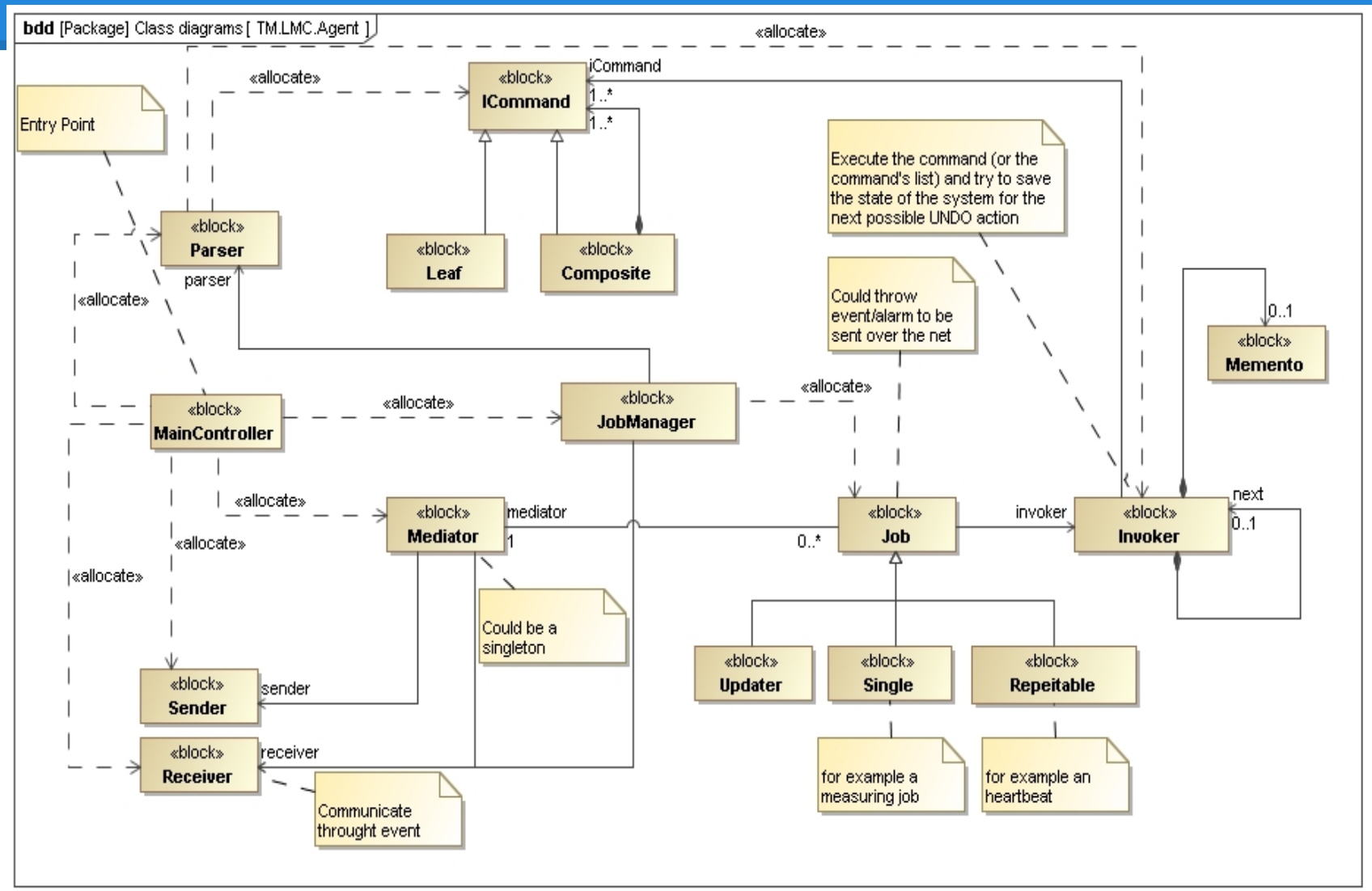
TM.LMC *specific*+ responsibilities *(LSR)*

- Preventing CPFs with TM
  - *if TM fails, the TM.LMC is still able to resolve and/or report the problem*
- Monitoring and control of
  - *TM process applications and process initialization*
  - *Execution and stopping order*
- Monitoring TM processes *(process status)* and hosts *(heartbeat)*
- Report TM element state for either the overall TM element or its components (or both)
- Configure (and support runtime re-configuration) for TM element and its components

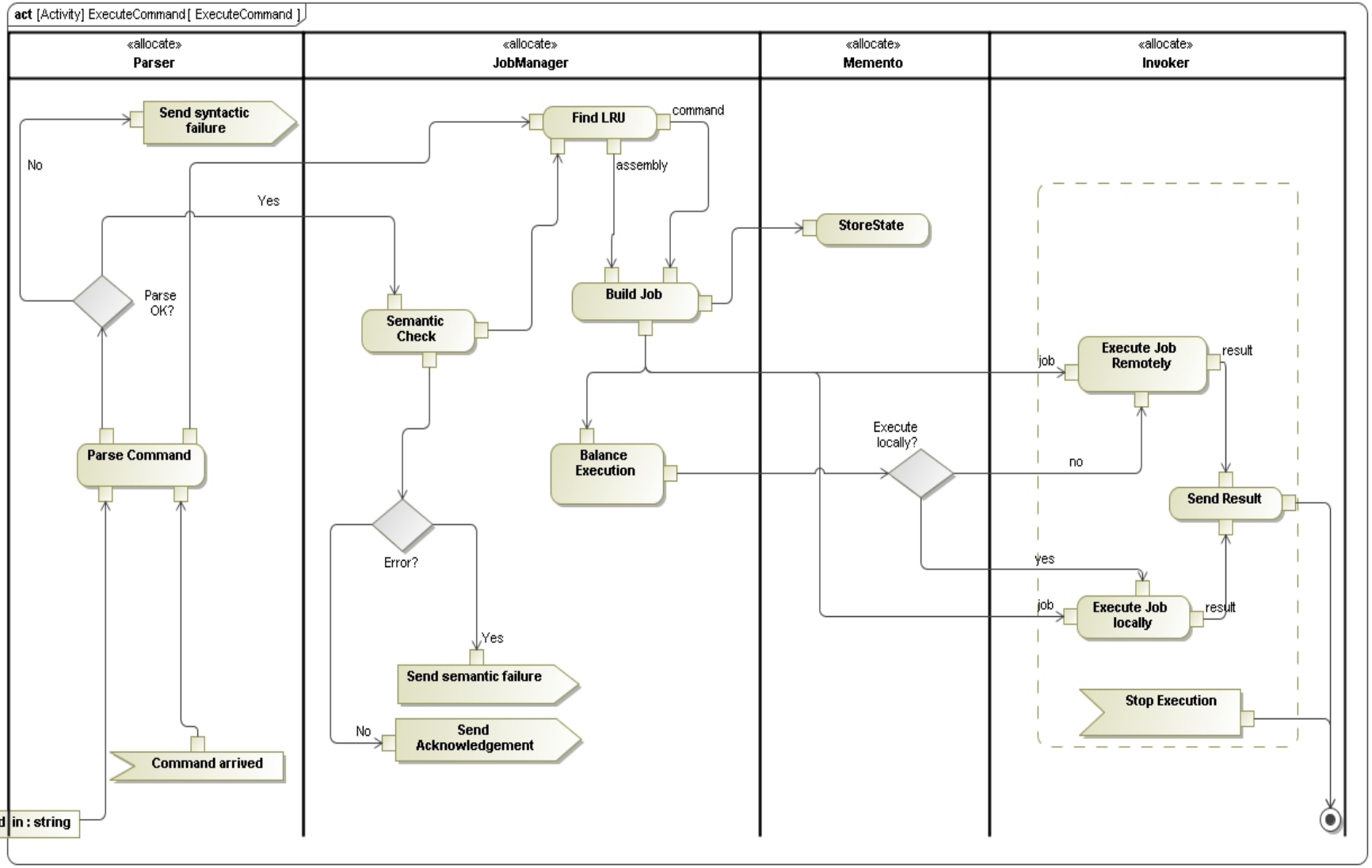
TM.LMC *functions*

- *Monitor TM status and report it to Operator*
- *Monitor TM performance*
- *Generate TM alarms*
- *Support TM remote diagnostics*
- *Generate TM failure indications*
- *Support TM software/firmware version control and upgrade*
- *Respond to detected TM sub-elements failures and alarms raised by any other sub-elements*
- *Configure TM sub-elements (for monitoring purposes)*.
- *Support TM lifecycle control*
- *Operate in case of sub-elements failures*
- *Provide direct access to monitoring data to external operators (engineers) in normal operations and in case of sub-elements failure*
- *Perform TM self-protection actions*

Architectural model



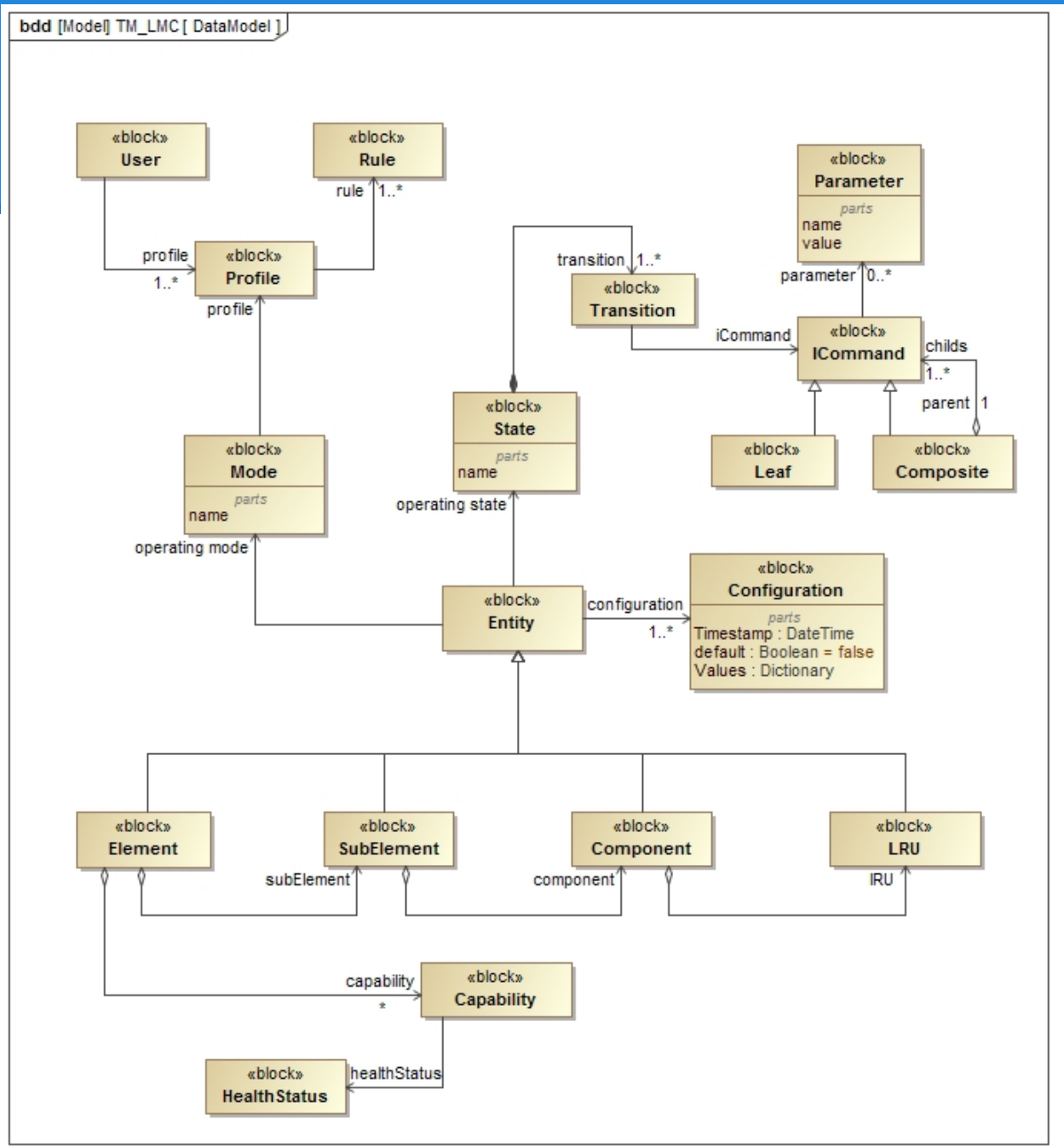
How does it work?



Monitoring and control process

- The TM.LMC monitoring and control process could be seen as an artificial intelligence problem
- Research in a state space (all the states reachable from an initial state).
- The state space is characterized by:
  - An initial state
  - A set of possible actions which transform a state to another one
  - A path from a state to another state (list of actions).

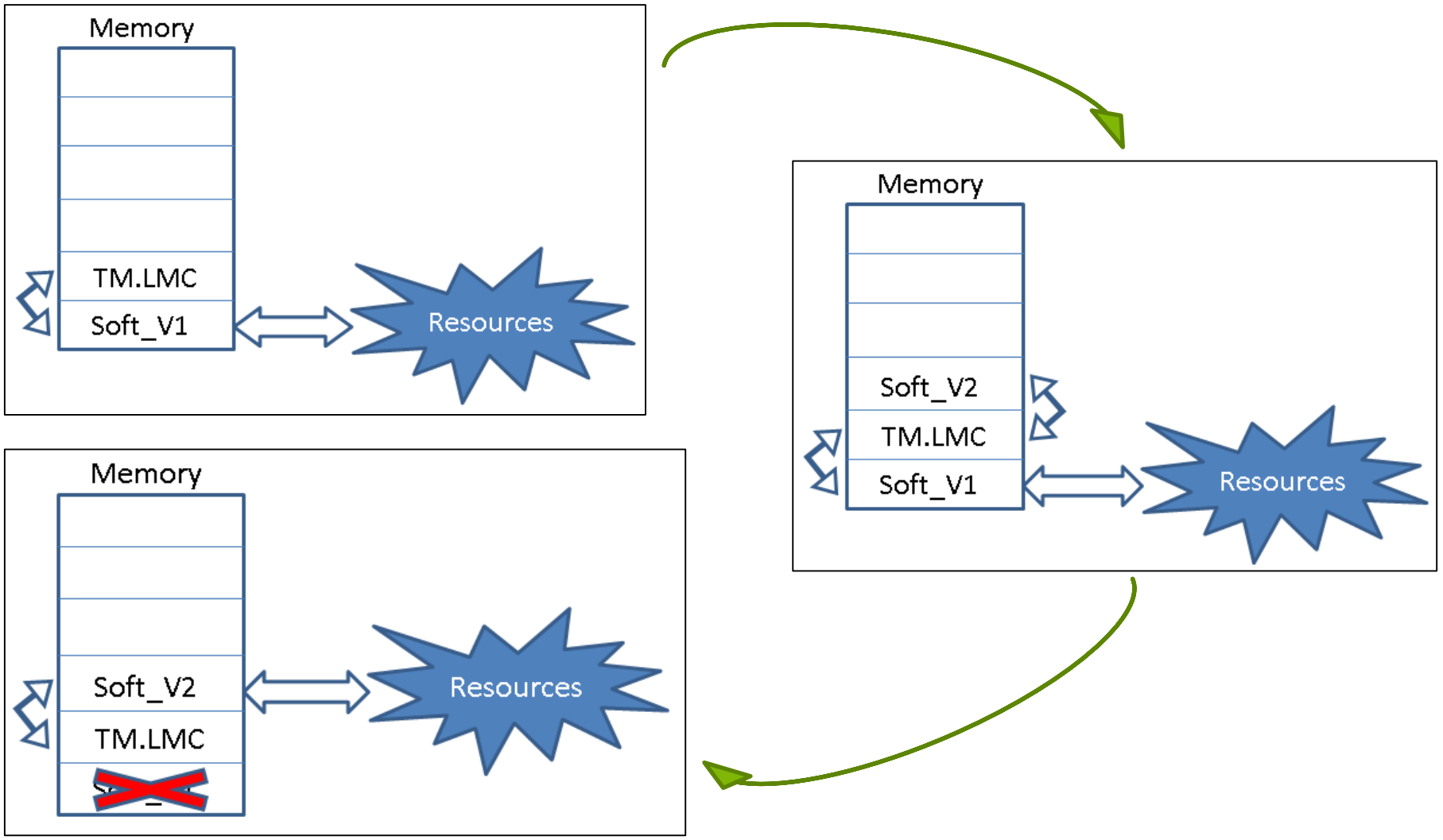
Abstract Data Model



LRUs and updating strategy

- LRU is defined as “an item that can be replaced on-equipment which does not require a special workshop to be replaced”
- *For TM.LMC, a LRU is a software assembly*
- The aim of updating strategy by TM.LMC: *replacing a piece of software **without** bringing to a service unavailability*

Updating strategy



Load balancing, scalability, reliability and availability

- Load balancing is a technique to increase the *scalability* and the reliability of architectures

- Cluster of worker nodes in the network which can do the same job (possibly coordinated)
  - *Scalability*: in case it is necessary, it is possible to add new server to the cluster
  - *Reliability*: the failure of one server does not affect the supply of the service (*fault tolerance*) to users.

- If the monitoring system is a cluster as well, it is possible to define the *high availability* (HA) system
  - Software layer to the application (number of active connections, geographic location, capabilities etc) or network level (Round-Robin DNS) of the ISO/OSI model

Common Framework Requirements

- **Maintainability and Support**
  - Object Oriented Design: The framework shall support an object-oriented/modular design of control components
  - Interoperability: The framework shall support interoperability among components developed in different programming languages (i.e. C++, Java, Python) both for server and client side

Common Framework Requirements

- **Communication Infrastructure**
  - Messaging service: The framework shall provide a messaging service/middleware, allowing distributed processes to communicate and exchange data
  - Publish/Subscribe communication: The middleware shall support the publish/subscribe pattern between communicating components

Common Framework Requirements

- **M&C Functionalities**
  - Keep-alive packets: The framework shall support exchange of keep-alive packets among components
  - Error reporting: The framework shall provide a mechanism for reporting communication faults as well as internal errors occurring in the control system components
  - Control GUI: The framework shall provide a graphical tool allowing the construction of instrument control GUIs

Common Framework Requirements

- **M&C Functionalities *(cont'd)***
  - Real-time Monitoring Tools: The framework shall provide a set of tools for real-time monitoring and plotting of monitoring data over time
  - Security Access: The framework shall have security access features to prevent unauthorized use or unintended access to the control system components
  - User categories: The framework shall provide features to manage user category levels

TM.LMC

Thank you !