Evolutionary JAD for more efficient business modeling

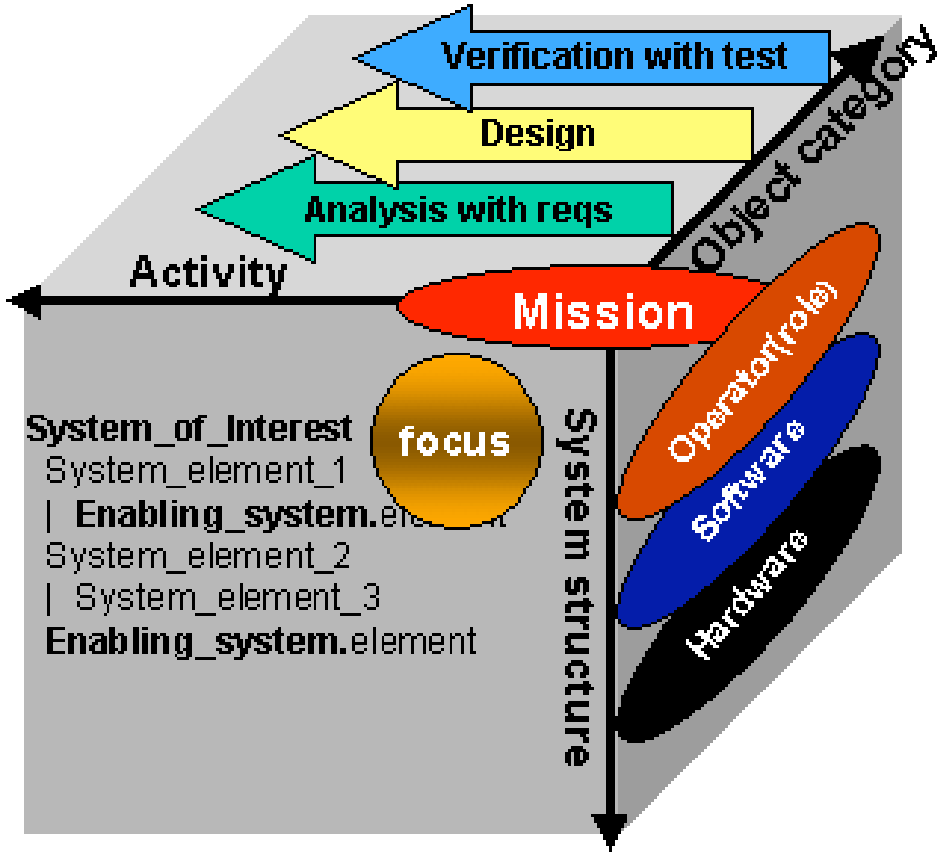
JAD = Joint Application Development

Ingmar Ögren, Romet AB

iog@romet.se

http://www.romet.se

The Mission is central

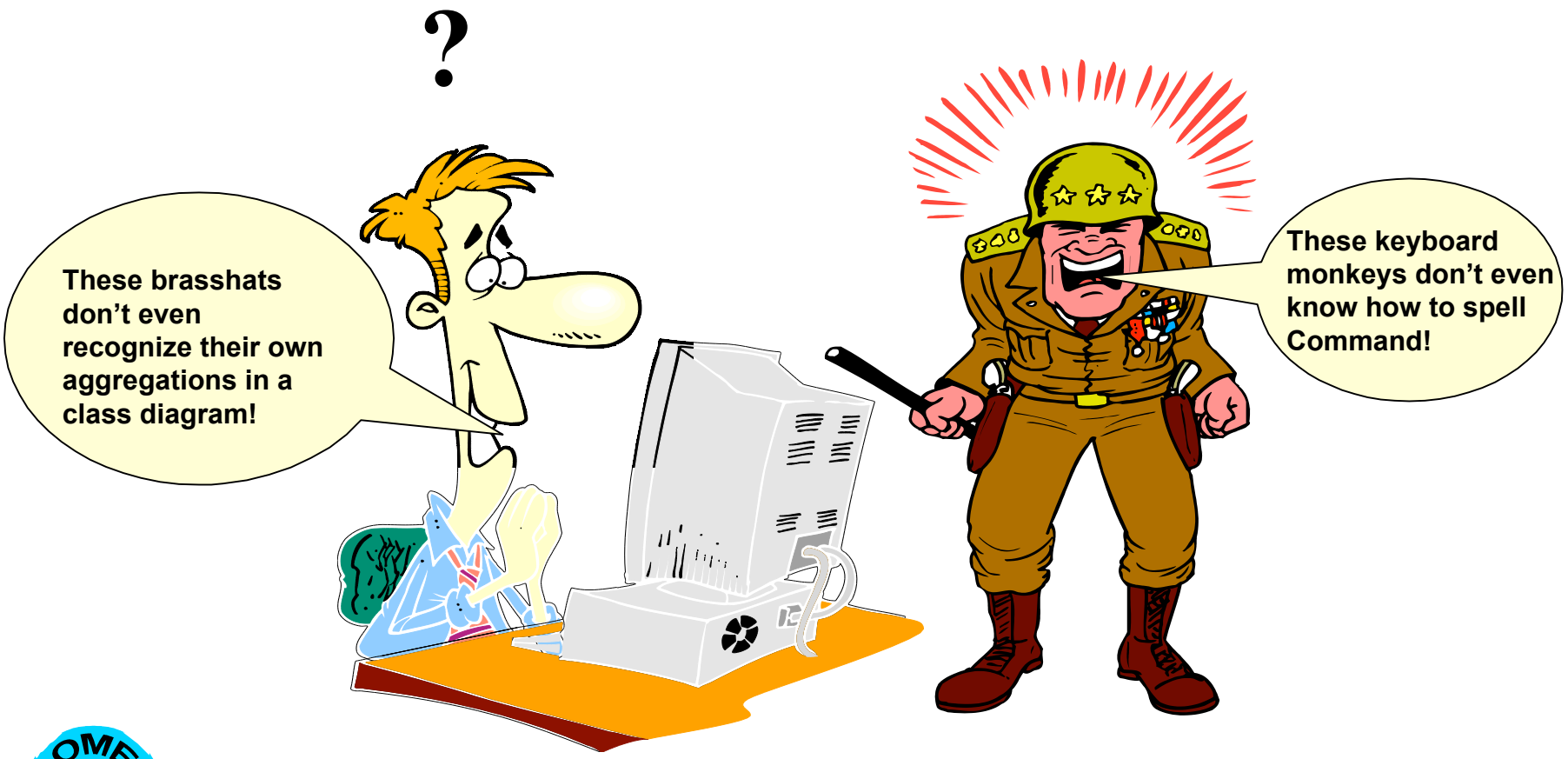


The systems engineering workspace is a three-dimensional space where you can change focus in the dimensions:

- **What you do**
- **Where you are active**
- **What object category**

Work in all three dimensions concentrates towards completion of Missions.

Communication is crucial



You cannot have a dialogue without mutual understanding

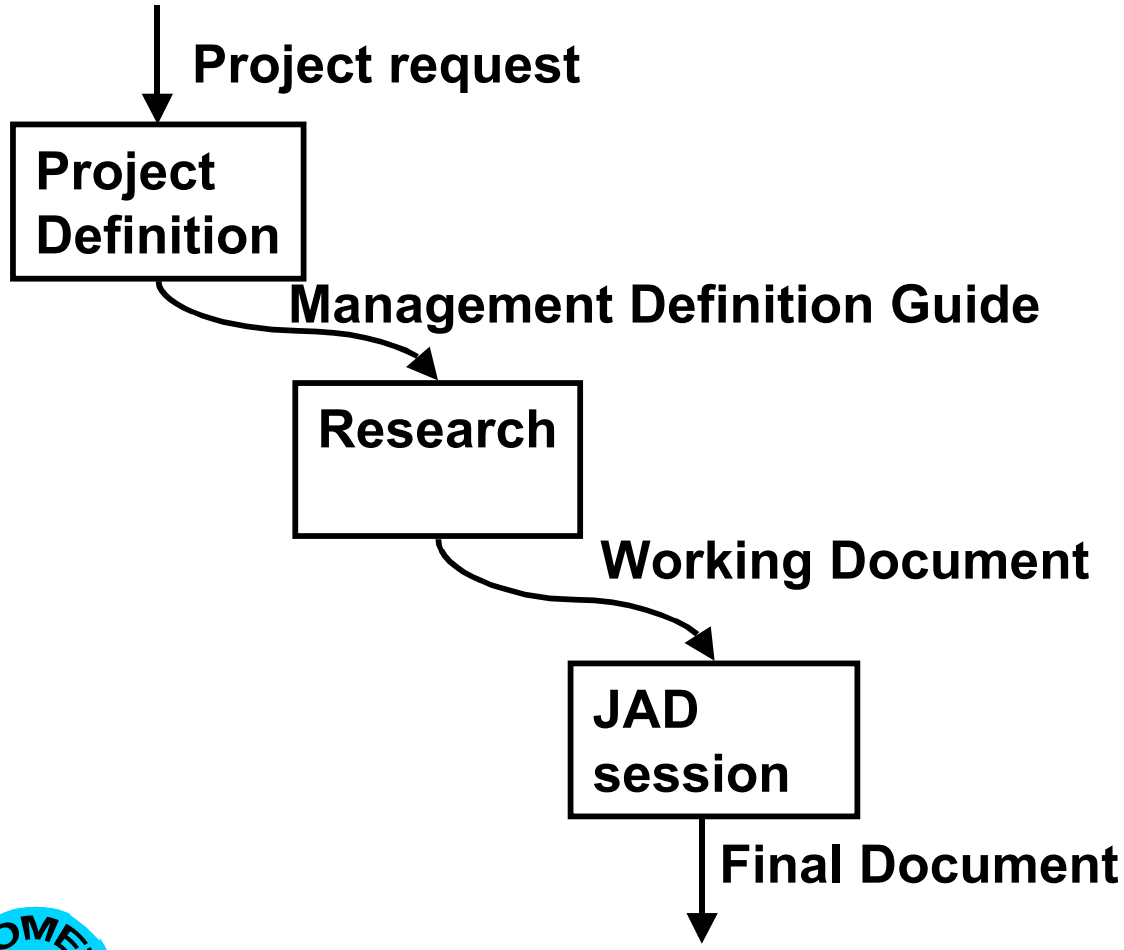
JAD manages communication

- Organized meetings with all stakeholders concerned
- Incremental development of system documentation
- Common view of the total system
- Close cooperation between application experts and IT experts
- Focus on management of open issues
- Joint design in real time of user interfaces, including logic and appearance
- Interleaved development of requirements and design
- Integrated verification of requirements and design
- Well proven record of success for more than 20 years

Proven record of ”best practice”

- Saves time, eliminates process delays and misunderstandings and improves system quality [Hol93].
- One of the best ways to reduce function creep, most of which results from poor initial requirements [Ant94]. Capers Jones claims this approach reduces function creep by 50%, and when used with prototyping, creep is reduced by another 10-25% [Str96b].
- By properly using transition managers, and the appropriate users, the typical cultural risk is mitigated while cutting implementation time by 50% [Eng96].
- Avoids bloated functionality, gold-plating, and helps designer's delay their typical "solution fixation" until they understand the requirements better [Whi95].
- Lays the foundation for a framework of mutual education, separate brainstorming, binding negotiation, and progress tracking [Whi95].
- Avoids the requirements from being too specific and too vague, both of which cause trouble during implementation and acceptance [Str96a].

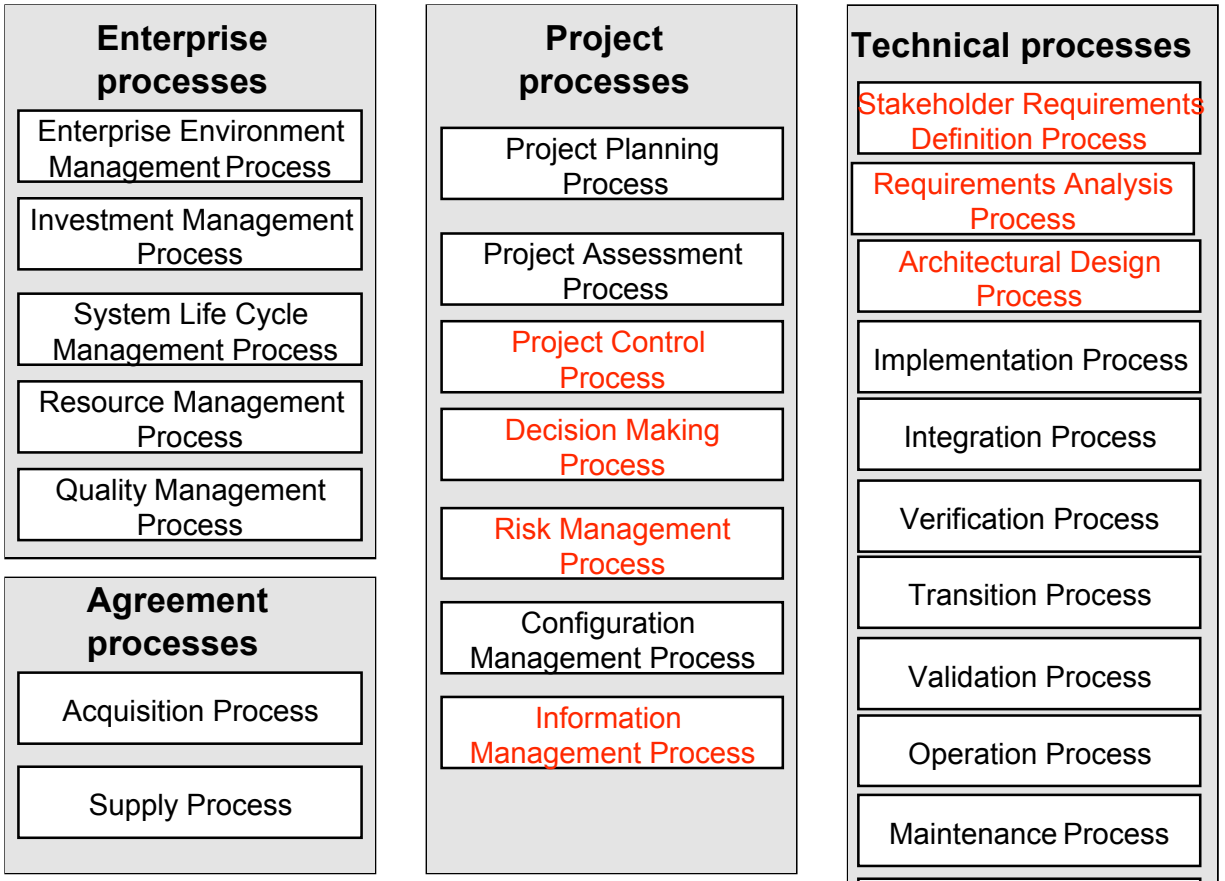
Phases of traditional JAD



Problems

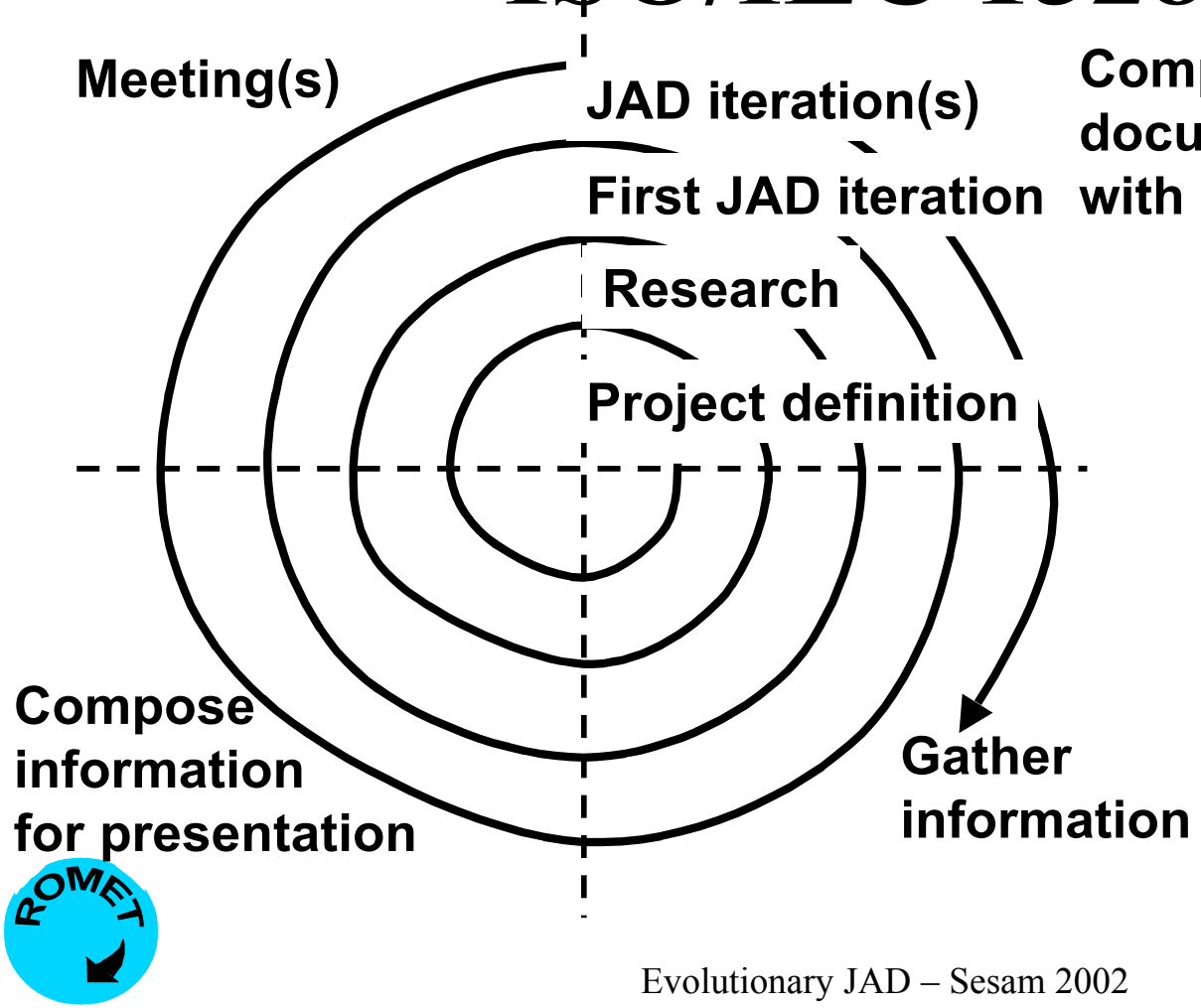
- phase orientation
- separate documents
- Data Flow diagrams
- doubtful compliance with ISO/IEC15288

Processes in the ISO/IEC 15288



Processes in red text are primarily addressed in JAD work

Updated JAD to comply with ISO/IEC 15288

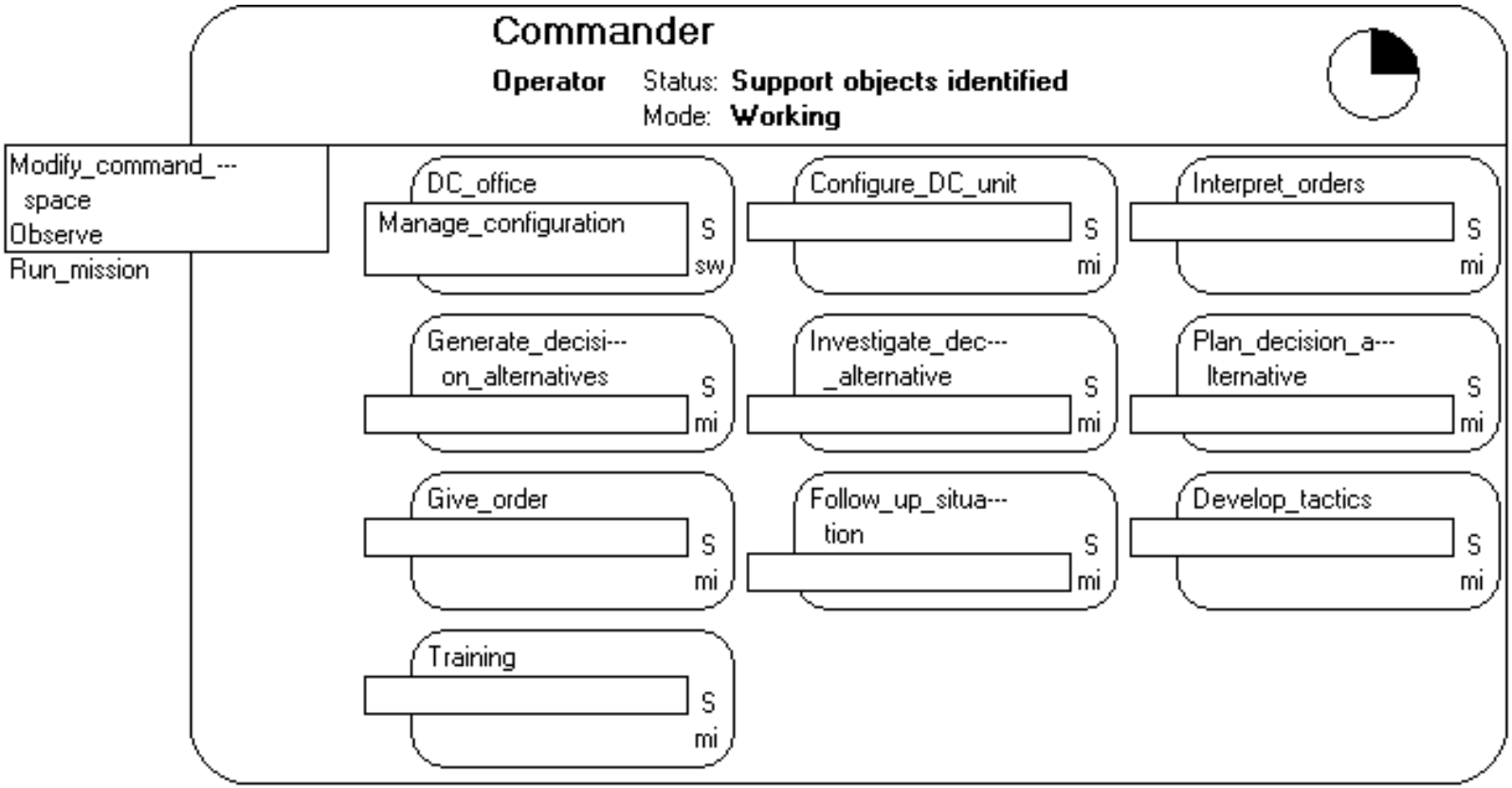


Listed JAD problems can be managed and Integrated Product Teams (IPTs) are supported

Simplify and extend the UML Component diagram

- **Allow object categories Mission, Operator, Software and Hardware**
- **Limit the diagram to two levels and get rid of the arrows**
- **Allow an alternative “Tree view” to visualize “Deep dependency structures” with distinction between inside/outside of the “system-of-interest”**

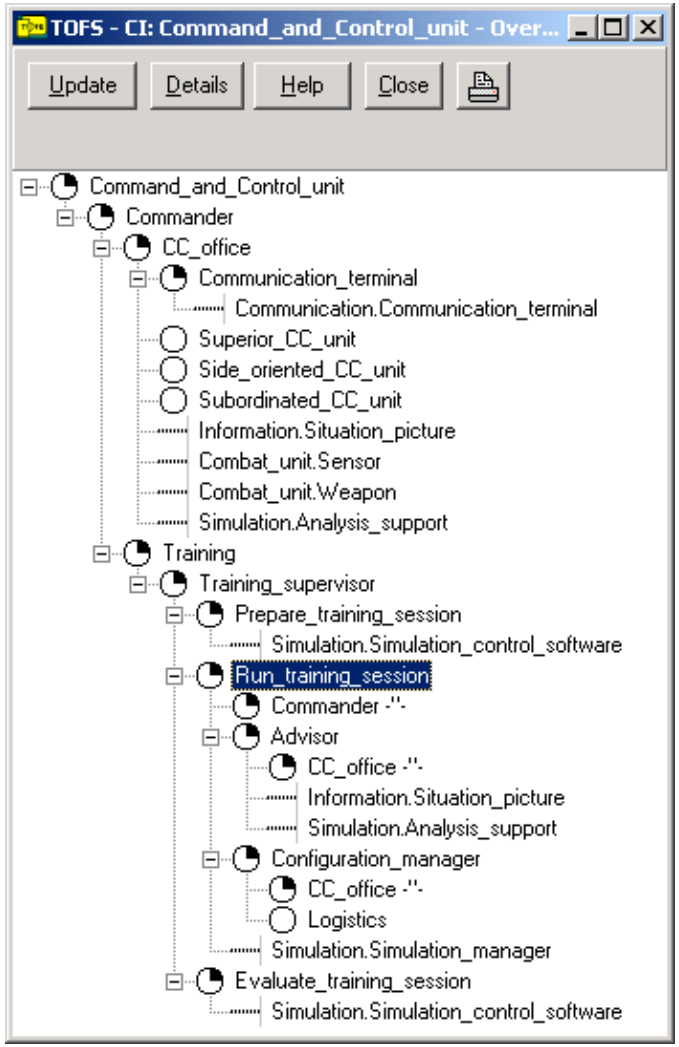
Simplified Component diagram



Trees for “deep structure”

The tree view shows:

- **Dependencies between objects (classes)**
- **Which objects are inside/ outside of the current “System-of-Interest”**
- **Which “Enabling systems” are required by the “System-of-Interest” to complete its missions**
- **Development status per object**



“Formal English” as a complement

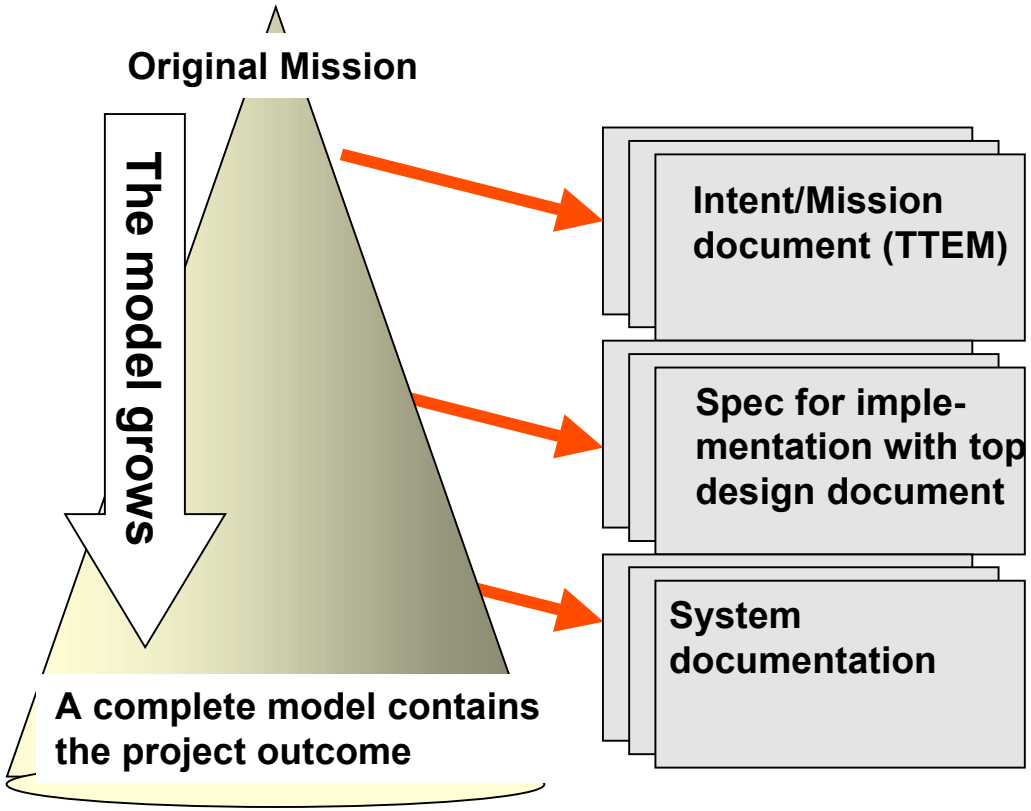
```
begin
        send Get_map(current_area)
        receive Present_map(current_area)
        receive Present_surveillance_resources(available_resources)
        for resource in 1..total_resource  loop
             send Select_resource(resource_id)
             receive Present_adjustment(possible_adjustment)
             send Adjust(current_adjustment)
        end loop
end

begin
      while Commander_messages_received  loop
            receive Get_map(area)
            Support_software.Get_area_map
            send Present_map(area)
            send Present_surveillance_resources(area)
            {The map information collected from the support software includes
            information about surveillance resources available in the area selected}
            receive Select_resource(Resource_id)
            send Present_adjustment(Possible_adjustment)
            {For each surveillance resource possible adjustment is received together
            with the map information}
            receive Adjust(Current_adjustment)
            Data_collection_own.Adjust_data_collection
            {The surveillance resource concerned is adjusted as required}
      end loop
end
```

Commander’s behavior

PMI software behavior

Documentation with MS Word



Summary

- **The combination of JAD, UML, IPTs and ISO/IEC 15288 represents state-of-the-art for system evolution**
- **Traditional JAD can be updated for evolutionary work with IPTs**
- **Management of Missions and 'Deep dependency structures" can be included in the UML**
- **UML can be simplified for stakeholder understanding**
- **With use of software tools, JAD sessions can be held with an IPT working directly with a system model presented on a data projector.**