SELEX
Sistemi Integrati
A Finmeccanica Company

Polo SOSIA
System Of System Intelligent Automation
ITALIAN CENTRE OF EXCELLENCE

System Of System Engineering Process

Selex Sistemi Integrati Company Experience

Genova, 17/07/2012

Contents

SELEX SISTEMI INTEGRATI - Mission

INTEGRATED SYSTEMS

HOMELAND PROTECTION

BORDER & TERRITORY PROTECTION
CRITICAL INFRASTRUCTURES PROTECTION
CRISIS MANAGEMENT MAJOR EVENTS

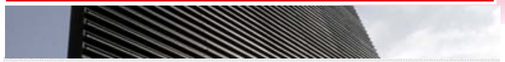
DEFENCE SYSTEMS

C4ISTAR SYSTEMS, NCW INFRASTRUCTURES
AIR DEFENCE SYSTEMS, BATTLESPACE,
C4ISTAR SYSTEMS

AIRBORNE, SURVEILLANCE & SECURITY SYSTEMS

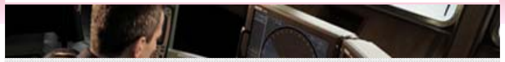
AIRBORNE MISSION SYSTEMS	ATC/ATM & AIRPORT SYSTEMS	VTMIS & MARITIME AWARNESS	ADVANCED I.T. FOR SECURITY, LOGISTICS AND AUTOMATION

SENSORS



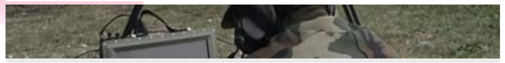
AVIONICS (EW, RADAR, EO)
NAVAL RADARS & FIRE CONTROL
SYSTEMS GROUND RADARS, ROCKET
LAUNCHER SYSTEMS

COMMAND & CONTROL



NAVAL COMBAT SYSTEMS
INTEGRATION
GROUND COMMAND & CONTROL
SYSTEMS

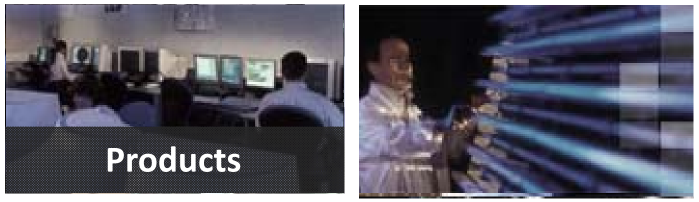
COMMUNICATION



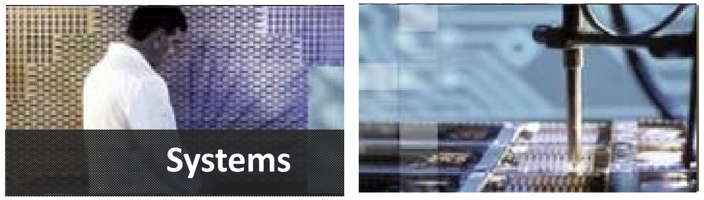
NAVAL & GROUND
AVIONIC CNI
PROFESSIONAL TETRA - WiMAX

SELEX SISTEMI INTEGRATI - Research & Development

During 2011 SELEX Sistemi Integrati invested more than 190 ML€ in R&D, about 25% of its turnover



- Active Electronically Scanned Array and Multi-Functional Radar
- Advanced Signal Processing, Algorithms and specific Functionalities
- GaAs/GaN Design & Production for Power and Wide Band Applications
- Microelectronics and Packaging
- Microwave and Photonic



- Analysis Simulation and Modeling of complex systems
- Distributed Sensors Networks and Data Fusion
- Network Centric Warfare Systems / Network Enabled Capability
- Middleware Development for Civil and Military Applications
- COTS maximization and integration, both HW and SW

SELEX SISTEMI INTEGRATI - Capabilities and Offers

Consolidated Capabilities

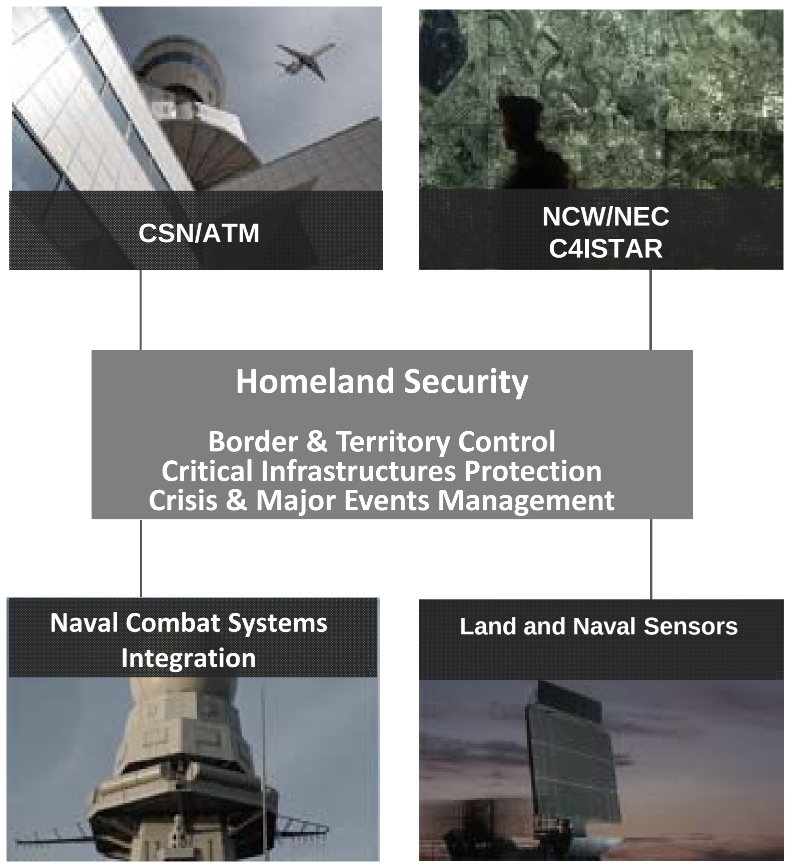
INTEGRATED SYSTEMS

- Naval Systems and Equipments for more than 100 naval units and 40 clients worldwide
- More than 300 ATM Systems and 1200 primary, secondary and meteo Radars provided to 150 Countries
- More than 25 Air Defense Centers worldwide
- VTMIS Italy, the largest worldwide, other systems in Yemen and Poland
- Homeland Security and Defense Integrated Systems in various countries

SENSORS

- More than 80 3D Long Range Radars
- Top supplier of LRR, NATO Class 1 Systems
- EMPAR Multifunctional Radar for Italian and French Navies, active and passive version
- KRONOS Radar New Family: 3D, active and multifunctional, for naval and land applications
- ATC En-route and Approach Radar
- Fire Control System Radar
- Coastal and Gap Filler Radar

Synergic Offer

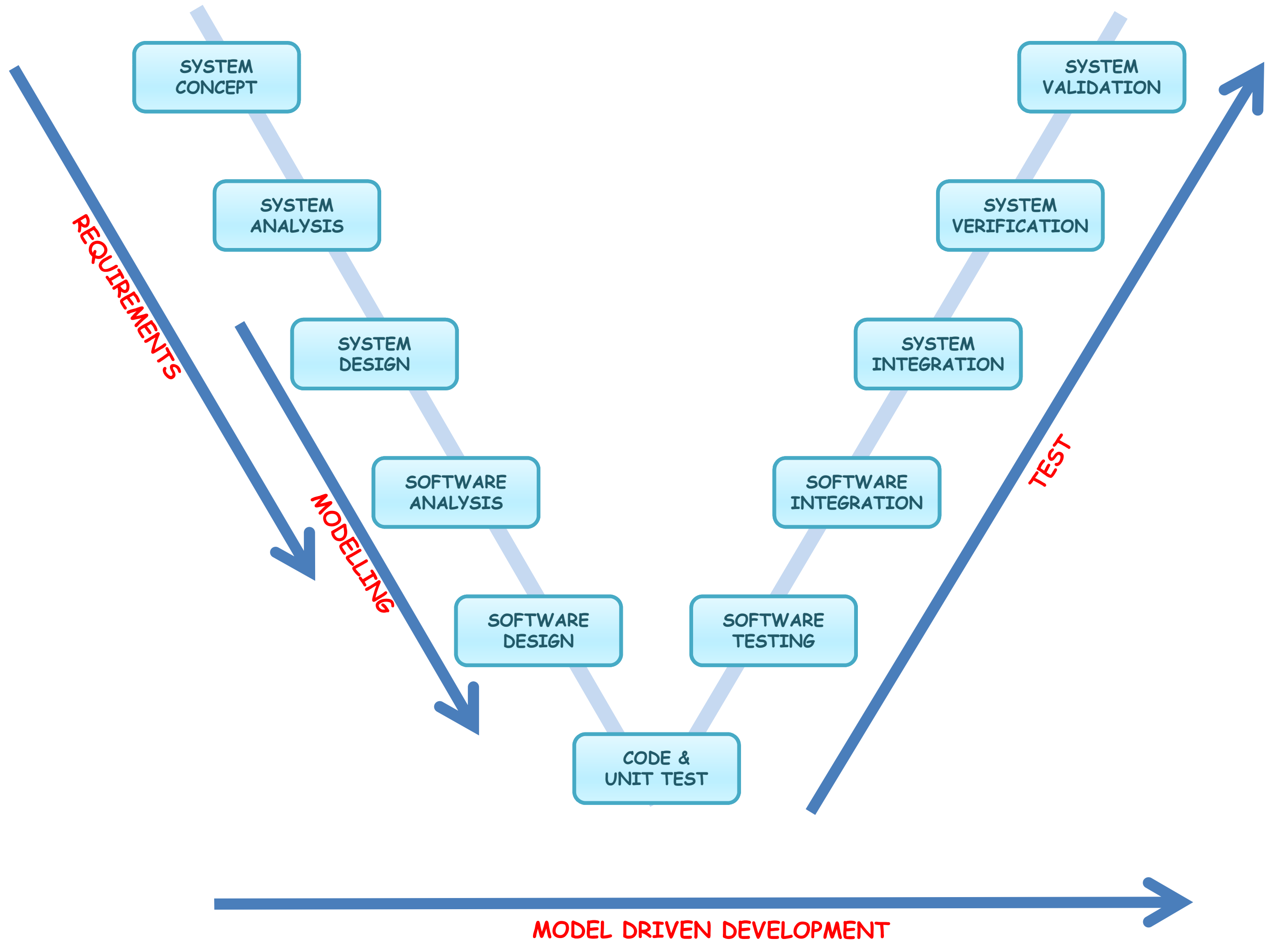

CSN/ATM
NCW/NEC
C4ISTAR
Homeland Security
Border & Territory Control
Critical Infrastructures Protection
Crisis & Major Events Management
Naval Combat Systems Integration
Land and Naval Sensors

Building up Systems Of Systems

- Systems Of Systems by Selex Sistemi Integrati
  - Naval Combat Systems
  - Integrated Airport and Air Traffic Management Systems
  - Homeland Defence Systems
  - Homeland Security Systems
  - Maritime Surveillance Systems
  - Enterprise Support Systems
- The realization of a System of Systems entails the integration of Company Systems and of Systems supplied by third parties
  - Systems by Selex Sistemi Integrati as building blocks for Systems Of Systems
    - Naval Combat Management Systems
    - Air Traffic Control Centres
    - Vessel Traffic Management Centres
    - C4I Systems
    - Air Defence Systems
    - Security Control Systems
    - Logistic Support Systems
    - Simulation and Training Systems
    - Gunnery Firing Control Systems
    - Navigation Systems
    - Radar Systems
  - Systems supplied by third parties include generally sensors, weapons and communication systems

SELEX
Sistemi Integrati

Recapture of System and Software Engineering Process



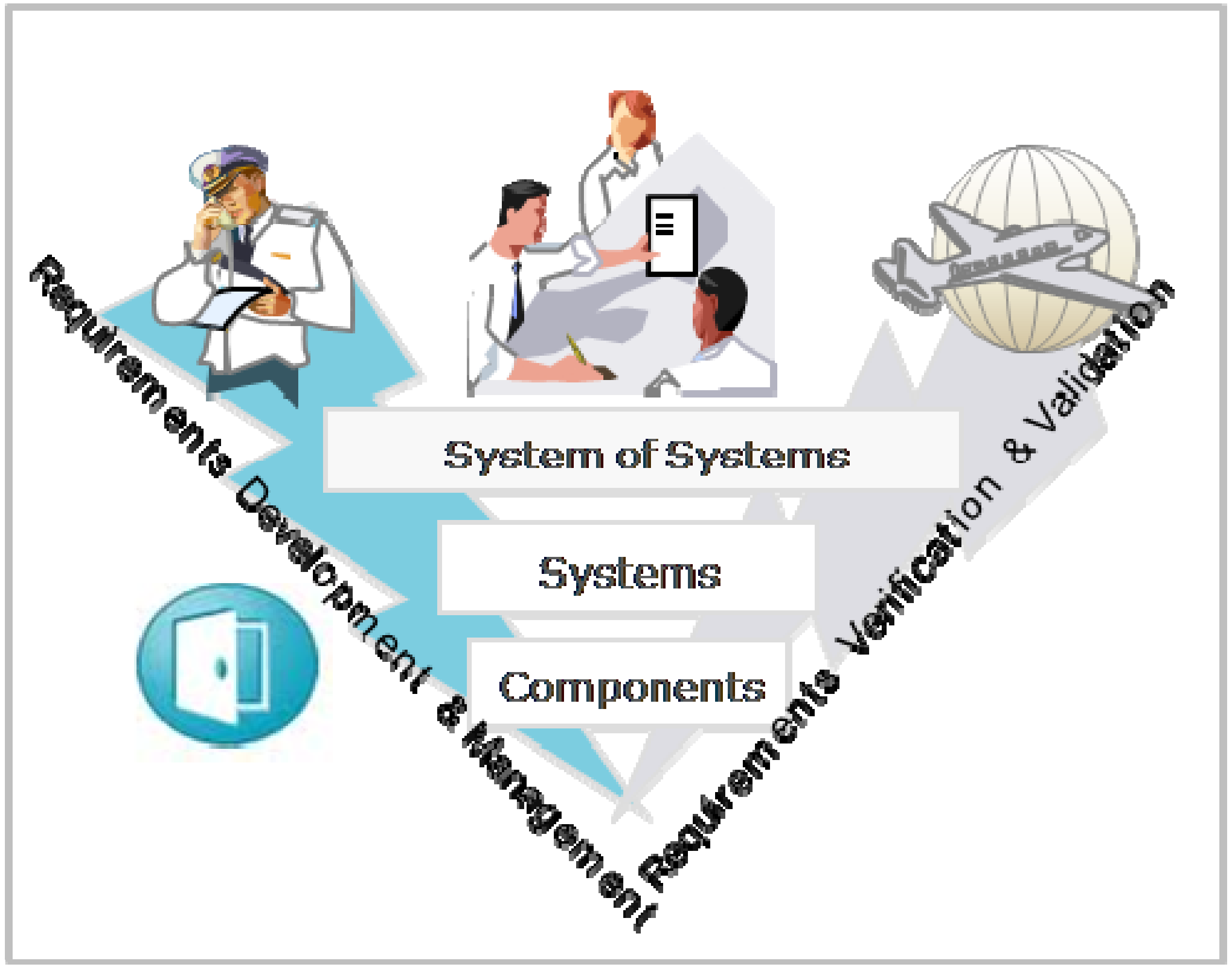
Requirement Issues for Systems Of Systems

- Realization of a System Of Systems is a very large scale project
  - Solutions for System Of Systems have to be largely based on systems and components reuse for affordability reasons
  - Reusability of requirements, properly developed and managed, is an important prerequisite systems and components reuse (System/Component Product Lines)

- Complexity of Systems Of Systems
  - Requirements to be structured in hierarchical levels corresponding to the system decompositions
  - Requirements to be traced across the different layers

- Requirements of System/Component Product Lines
  - Requirement expressed fairly independent from the Systems Of Systems where they are employed
  - Requirement factorisation to achieve adequate level of requirement abstraction and granularity
  - Requirement customization to meet specific project needs

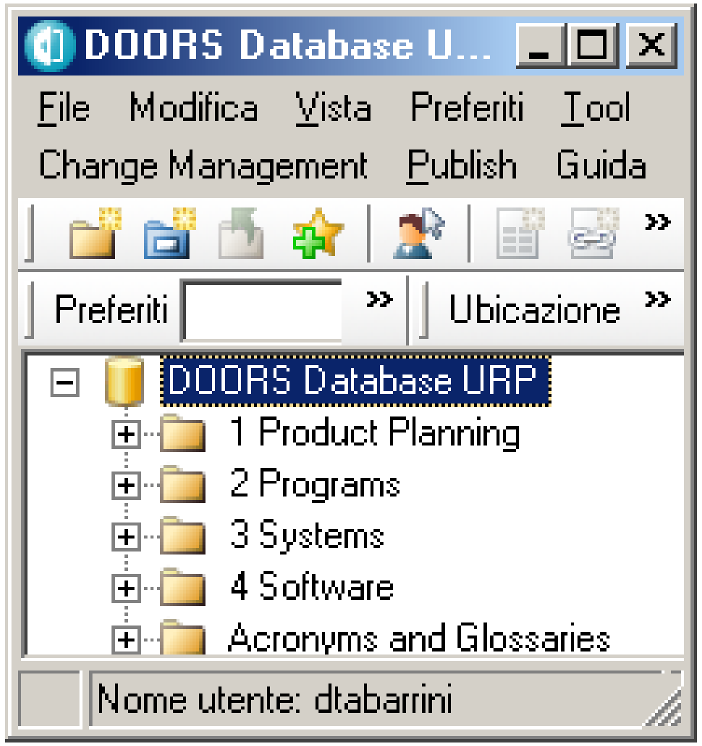
Requirement Life Cycle


Requirements Development & Management
Requirements Verification & Validation
System of Systems
Systems
Components

Company Unified Requirement Platform

- Unified Requirement Platform enabling:
  - management of different layers of requirements
  - identification of System/Component Product Lines requirements for sharing and reuse across different projects
  - links with customized requirements

- Unified Requirement Platform structured into:
  - Product Planning Requirements
  - Program Requirements
  - Customer Requirements
  - Systems Of Systems Requirements
  - System Requirements
  - Component Requirements



DOORS Database URP
1 Product Planning
2 Programs
3 Systems
4 Software
Acronyms and Glossaries

Nome utente: dtabarrini

Requirement Development Activities

- Requirements analysis started from the definition of operational concepts and scenarios
  - based on all the available information about the customer needs

- Requirement analysis techniques
  - elaboration of use cases
  - modelling the expected behaviour through UML or SySML diagrams (e.g. use case diagrams, state diagrams, activity diagrams, etc.)
  - simulation of system functionalities (capturing the essential algorithms characterizing the system)
  - user task analysis (identifying all roles and tasks of the system users)
  - prototyping the user interfaces (defining all the interactions between the system and the user)

System Requirements Development

- Requirement of System of Systems defined through analysis techniques to meet Customer Requirements
- Technical solution for System of Systems identifying the composing Systems used as building blocks
  - Requirement analysis and development for each composing System
  - mixed top-down and bottom-up approach to ensure requirement compliance and realize competitive solution s
- Trade-off Requirements of Systems used as building blocks
  - Maximizing requirements already developed and verified in System Product Line
  - ensuring the compliance with the allocated System of Systems Requirements
    - or Customer requirements that are directly allocated if the System is provided as stand-alone solution
- Key drivers for System Requirements
  - Product planning policy to enable system product reuse
  - Requirement engineering activities to enable requirements reuse
  - Requirement quality to ensure understandable, flexible, expandable, customizable and testable formulation

Components Requirements Development

- Component Requirements analysis and development following the same approach of System Requirements
  - mixed top-down and bottom-up approach to ensure requirements compliance and to realize competitive solutions
- Inputs for Component Requirements
  - System Requirements allocated to the Component
  - Component Requirements that a Component Product Line is already able to ensure
- Trade-off Component Requirements
  - maximizing the requirements already developed and verified in the Component Product Line
  - ensuring the compliance with the allocated System Requirements

Requirement Management Activities

- Activities of Requirements Management conducted along the whole lifecycle to ensure that:
  – requirements are properly understood and documented
  – commitment to requirements is obtained from all the stakeholders (Customer, designers, developers, suppliers, etc.) according to the requirement relevance
  – changes of requirements are properly managed
  – bidirectional traceability is kept between:
    - Customer Requirements and System Of System Requirements
    - System Of System Requirements and System Requirements (or between Customer Requirements and System Requirements if the System is provided as stand-alone solution)
    - System Requirements and Component Requirements
  – alignment between project work and requirements is ensured

Requirements Management Attributes

- Requirement specification structure has been standardized including the definition of a common set of requirement attributes
  - *Identifier* – unique requirement identifier in the Unified Requirement Platform
  - *External Identifier* – requirement identifier for publishing in the applicable project documents
  - *Project Application* – applicable project for the requirement
  - *Title* – heading of the requirement
  - *Object Text* – textual description of the requirement;
  - *Capability/Sub-capability* – functionalities specified by the requirement
  - *Type* – requirement type, e.g. functional, performance, non functional, etc.
  - *State* – state of the requirement life cycle
  - *Allocation* – subsystem or component where the requirement is allocated
  - *Verification Level* – requirement verification level, e.g. system, subsystem, component
  - *Verification Method* – requirement verification method, i.e. test, analysis, inspections, demonstration
  - *Verification Test* – link to verification test case
  - *Version* – requirement versioning number
  - *Implementation Build* – system release implementing the requirement for the first time

Requirement Development for System/Component Product Lines

- Definition of the System/Component Product Lines baseline requirements
  - starting from background of requirements established for the main projects carried out by in the different domains
  - demanding great efforts of requirement refactoring

- Requirement refactoring
  - capturing all essential features that are likely applicable across multiple projects
  - requirements expressed in a fairly independent way from the system context
  - adequate level of abstraction and granularity
  - widest application in multiple projects minimizing the needs of customization

- System/Component Product Lines baseline requirements managed in the Unified Requirement Platform
  - linked to the customized requirements which may correspond either to
    - requirement customizations for specific project needs
    - originating project requirements before refactoring (to keep track of the originating requirement)

Software Architecture Modeling Aims

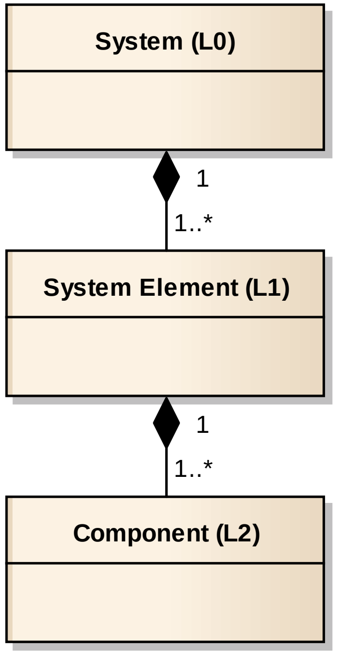
- The amount of software required for Components and Systems building up Systems Of Systems is increasingly growing along with the system complexity
  - => the software architecture of the constituting parts of the SoS is a key aspect of the system design process

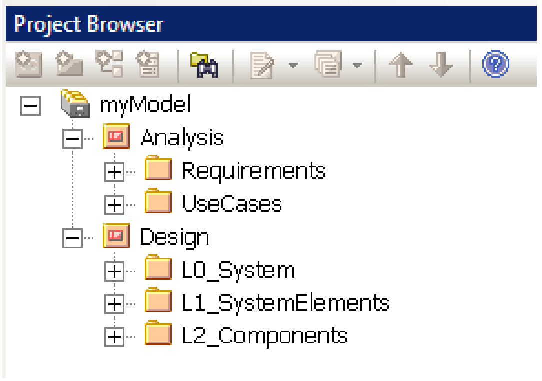
- Software architecture can be dramatically complex for large systems
  - => modeling has a major impact on software engineering and is critical for the success of every enterprise-scale solution

- Reference architecture models inspiring design of concrete architectures for specific projects
  - – generic and somewhat abstract blueprint-type view of a system
    - includes the system's major components, the relationships among components, and the externally visible properties of those components
  - – domain solutions generalized and structured to achieve reference architectures harvesting patterns observed in successful implementations
  - – not usually designed for a highly specialized set of requirements but starting point to be specialized according to requirements

Software Architecture Modeling Guidelines

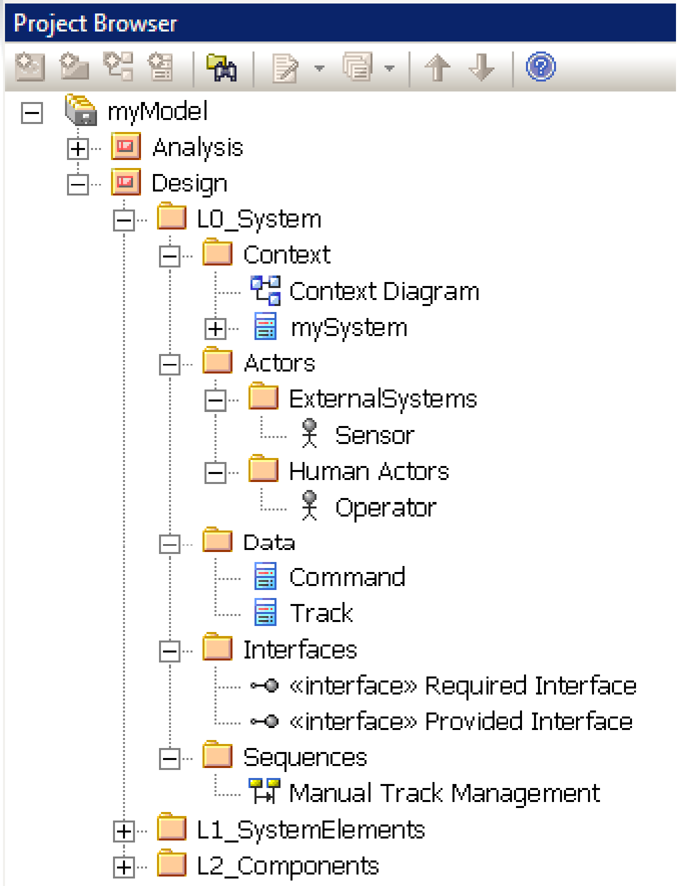
- Guidelines developed and applied to ensure a homogeneous UML modelling approach
- Architecture modelling framework decomposing the system architecture into different levels referenced as L0, L1, L2,…
  - Each layer is totally independent from the lower ones and it is the refinement of the upper one
  - L0 represents the System Level
    - i.e. the System used as building block of Systems Of Systems
  - L1 represents the System Element
    - i.e. major elements that constitutes a system at the first level of decomposition
  - Ln represents the Component Level
    - in most of practical cases two levels have been identified for the Component, L2 and L3
  - Software architecture modelling framework at system level does not represent internal design of components
    - more suitable to have a separate architecture model for each component identified as leaf of the system decomposition



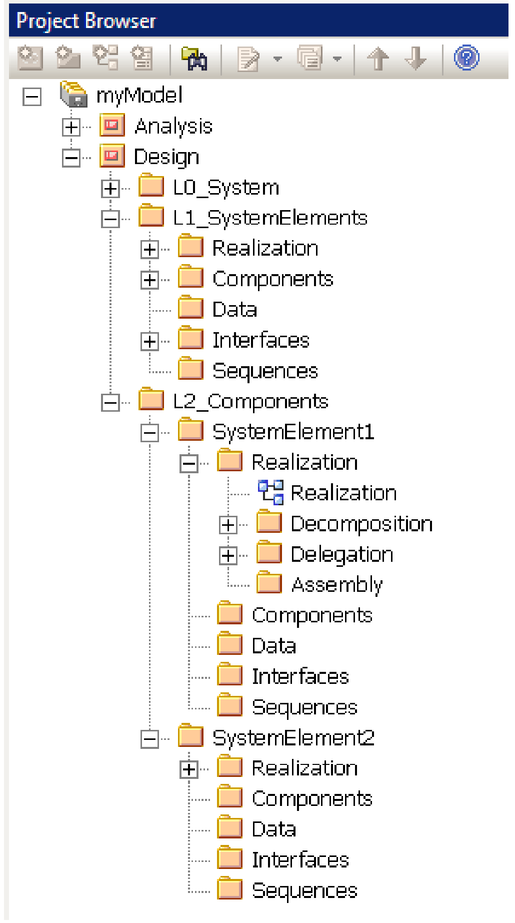


```
myModel
  Analysis
    Requirements
    UseCases
  Design
    L0_System
    L1_SystemElements
    L2_Components
```

Software Architecture Modeling Levels



```
Project Browser
myModel
  Analysis
  Design
    L0_System
      Context
        Context Diagram
        mySystem
      Actors
        ExternalSystems
          Sensor
        Human Actors
          Operator
      Data
        Command
        Track
      Interfaces
        «interface» Required Interface
        «interface» Provided Interface
      Sequences
        Manual Track Management
    L1_SystemElements
    L2_Components
```



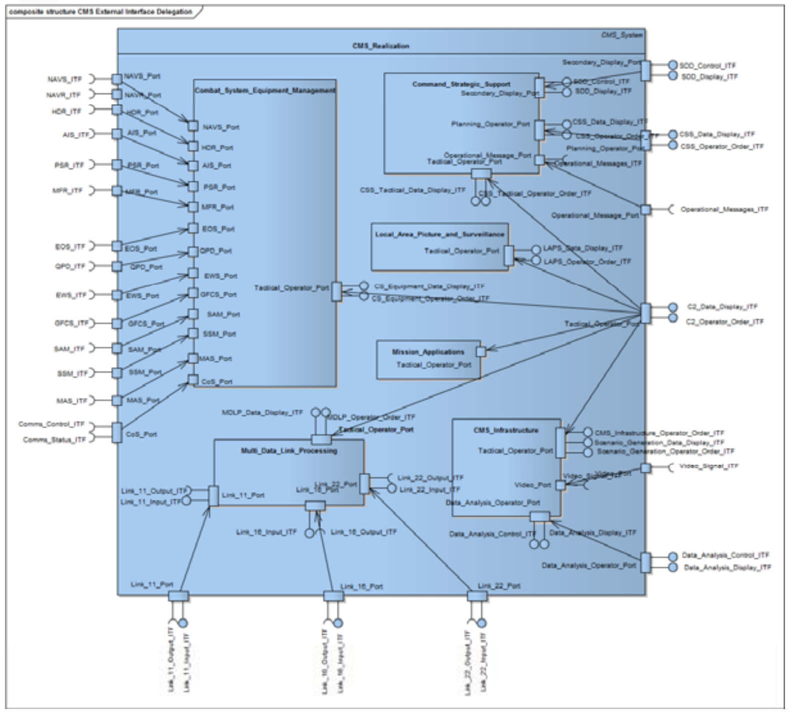
```
Project Browser
myModel
  Analysis
  Design
    L0_System
    L1_SystemElements
      Realization
      Components
      Data
      Interfaces
      Sequences
    L2_Components
      SystemElement1
        Realization
          Realization
          Decomposition
          Delegation
          Assembly
        Components
        Data
        Interfaces
        Sequences
      SystemElement2
        Realization
        Components
        Data
        Interfaces
        Sequences
```

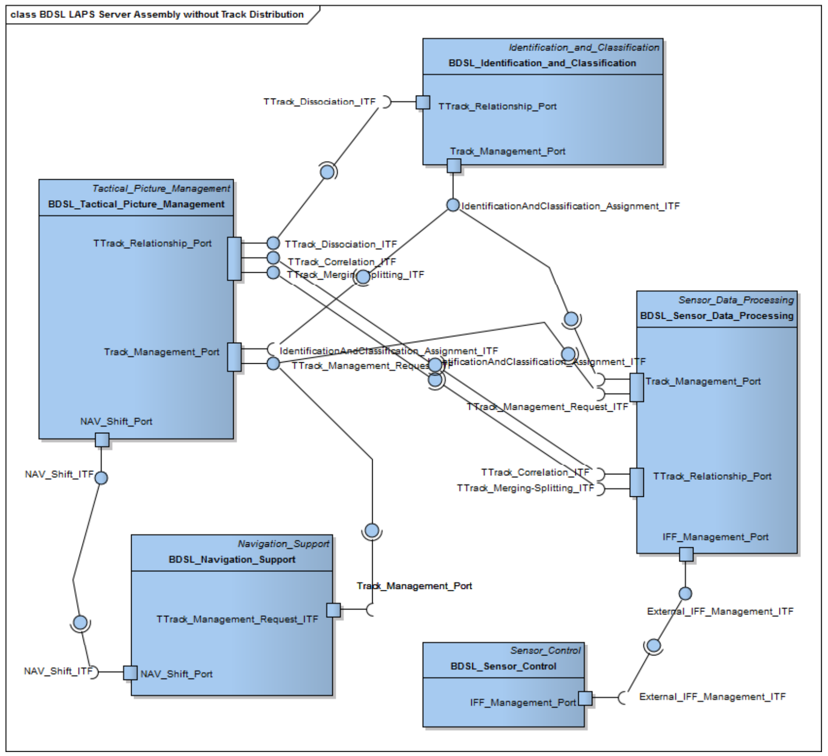
Software Architecture Modeling Views

- Architectural models organized by views to answer three specific concerns
  - What are the constituent parts of a system?
  - Who is responsible to implement an interface?
  - How do the constituent parts of the system interact?

- Modelling views included in the Realisation package of each modelling level
  - Decomposition, one or more diagrams to model the decomposition of an item into its constituting parts
  - Delegation, one or more diagrams where each external interface of an item is delegated down to its constituting parts
  - Assembly, one or more diagrams to model the interactions among the constituting parts of an item

Examples of Delegation and Assembly Views





Interface Modeling

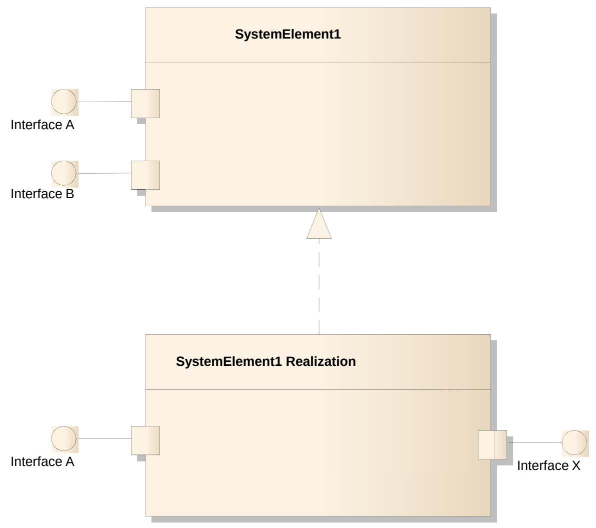
- System architectural model applying the concept of "design by contract"
  - server offering services and public methods to exploit the services
  - clients invoking the methods of the server class
  - contractual clauses are established by the server and are invariant from the client
    - establishing the conditions to execute the service (e.g. signature consisting of the arguments to be passed and the range to be respected by the arguments)
  - interface as semantic tool to stipulate a contract between a server and any client
    - if an object implements an interface it ensures to any client the implementation of all the public methods specified by the interface representing the services offered by the object.

- Architecture Model ensures that all the contracts between software applications are represented through the interfaces in complete and consistent way
  - all system interfaces identified in a single model

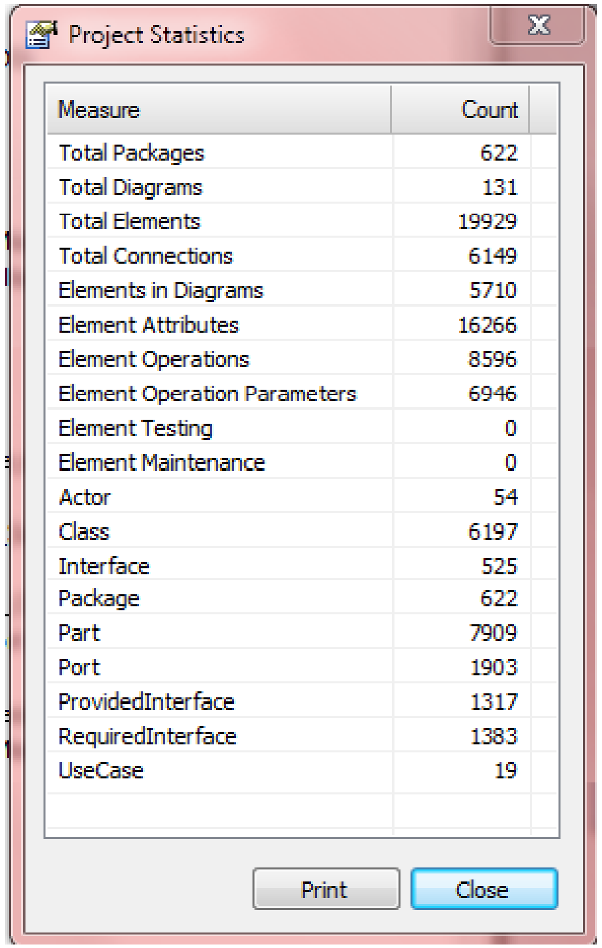
- Interface Requirement Specifications can be derived by Architecture Model
  - detailing all the services to be provided by each software application to ensure the overall functionalities at system level

Specialization of Reference Architecture

- Reference Architectures applicable at system domain level
- Reference Architectures to be specialized for a single project
  - to take into account specific requirements, context, actors, required interfaces, required functionality and system configuration
- Project specific architectural models defined by inheritance of the modelling elements of the Reference Architecture model
- Project specific architectural models customized following two main paths
  - deactivate existing elements
  - add new elements



Sample Statistics



Measure	Count
Total Packages	622
Total Diagrams	131
Total Elements	19929
Total Connections	6149
Elements in Diagrams	5710
Element Attributes	16266
Element Operations	8596
Element Operation Parameters	6946
Element Testing	0
Element Maintenance	0
Actor	54
Class	6197
Interface	525
Package	622
Part	7909
Port	1903
ProvidedInterface	1317
RequiredInterface	1383
UseCase	19

Statistics of the UML model (work still in progress) representing the reference architecture of the Naval Combat Management System Domain.

Model Driven Development

- Model Driven Development assisting design documentation production
  - applied for System Design Description, Interface Requirement Specification, Interface Design Description, Software Design Description
  - automatically taking models and their related diagrams and descriptions
  - automatically applying documentation structuring rules
  - easing consistent documentation quality

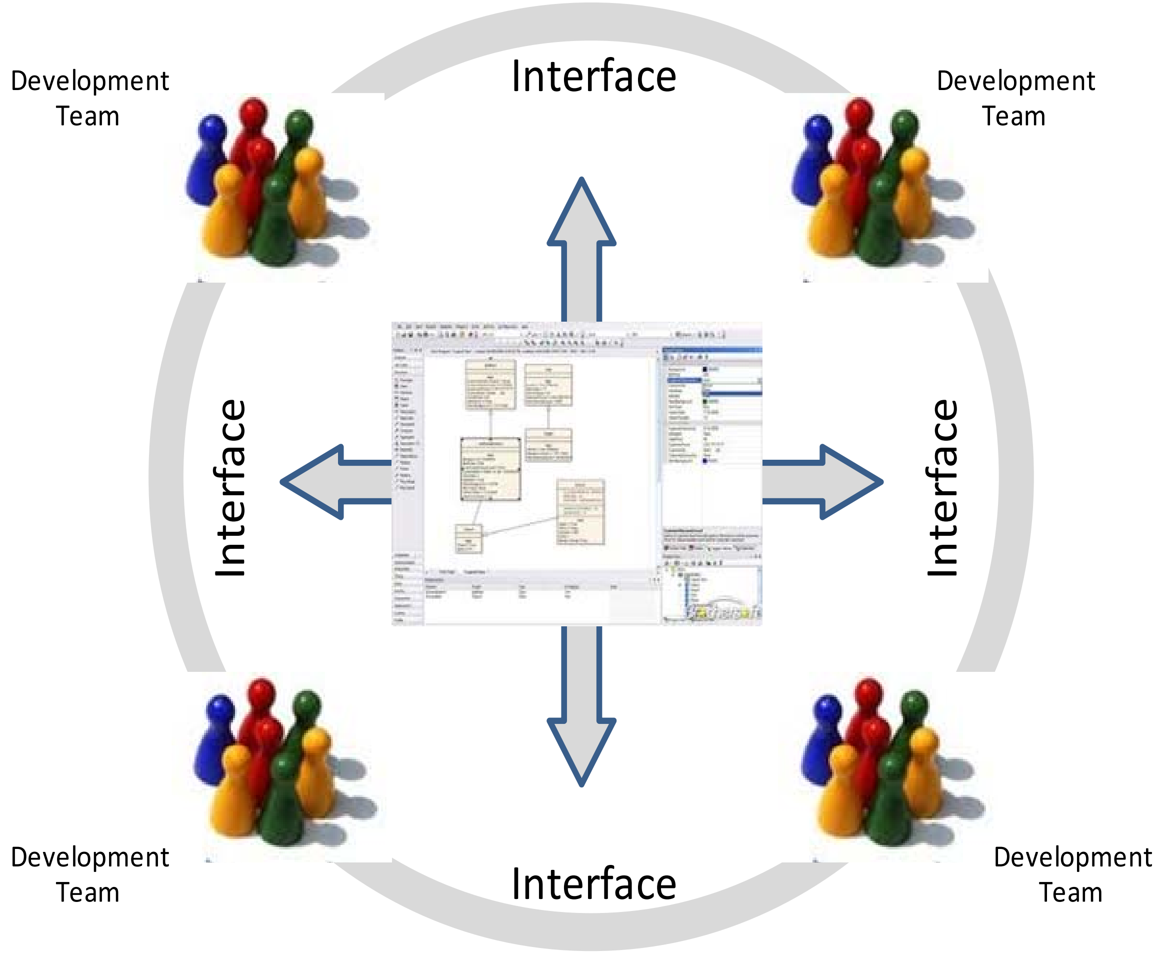
- Model Driven Development assisting code production
  - applied for interface implementation code
  - enabling permanent documentation/code/model consistency

- Model Driven Development assisting through-life-cycle support
  - easing quality metrics
  - facilitating the impact assessment of evolutions

Model Driven Realization of Systems Of Systems

- Systems Of Systems are characterized by tremendous efforts of software development
  - large number of teams and software engineers
- Design models and software requirements are the main inputs for software implementation
- Software development teams have to share information also about the actual implementation of the interfaces
  - information automatically generated by the system design model
- Model Driven approach assisting also integration, deployment and maintenance of Systems Of Systems
  - producing configuration files
  - generating interface design information to feed testing and integration tools

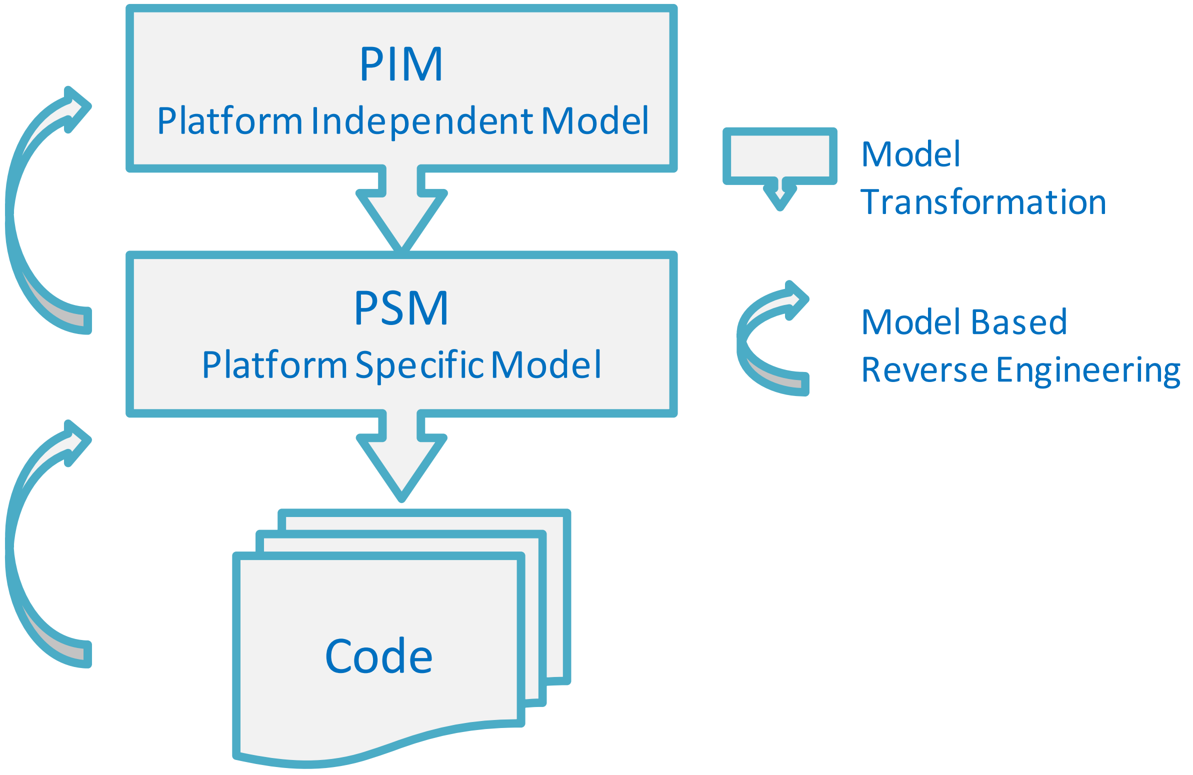
Development Scenario of Systems Of Systems



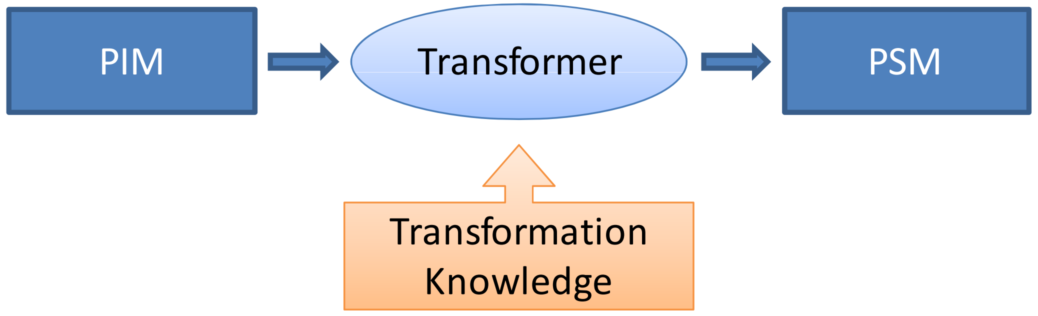
Example of development complexity

- Large System that imply the development of 20 software applications (examples are given by Naval Combat Systems):
  - average of 20 software development teams working in parallel
  - each team with an average of 10 software engineers
    - covering different roles, from the software application project leader to software analysts, designers, programmers and testers.
  - more than 200 software engineers are employed to develop a large software system

Model Transformations

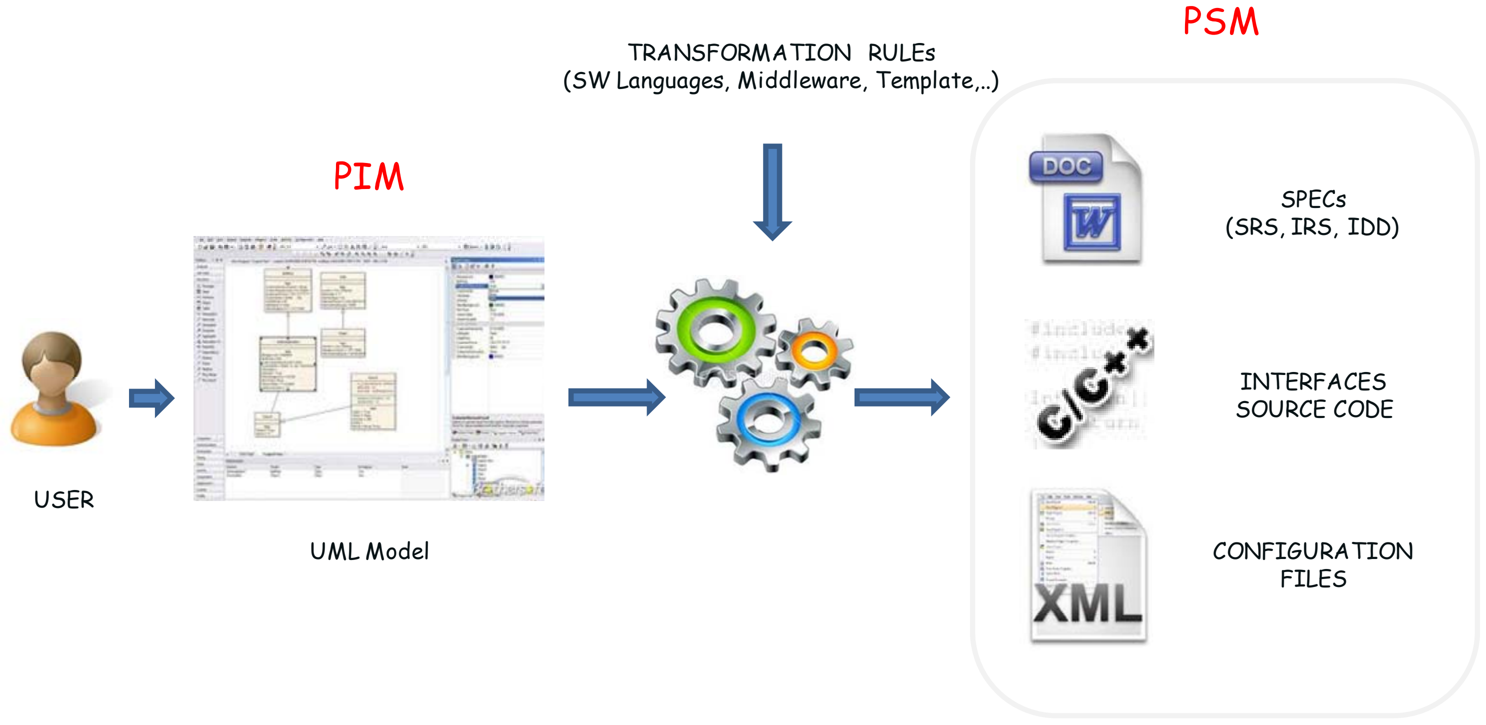


PIM-to-PSM Transformations



- Platform Specific Models based on the middleware features supporting the realization of interfaces and the deployment of software applications
- Middleware products developed by SELEX Sistemi Integrati
  - MIDAS, able to support C++ applications, for Naval Combat Management Systems
  - CARDAMOM, able to support both C++ and Java applications, for Air Traffic Management Systems
  - Servizi della Digitalizzazione, able to support Java applications, for C4I Systems
- PIM-to-PSM transformations are to map interface specification and design defined in the PIM into interface implementation defined in the PSM
  - An interface is to be implement exploiting middleware services (e.g. Messaging, Data distribution, remote procedure calls, ...)

PIM-to-PSM transformation chain and artefacts



Generation of Interface Design Description and Source Code

- Interface Design Descriptions and Interface Source Code can be automatically generated from the system architecture model
  - the interface design information in the model fully define the implementation of services provided by a system element with associated methods and data
- Interface Design Descriptions document the description of all the data associated to an interface (e.g. message fields, procedural call parameters) with enough accuracy to enable software coding
- Interface Source Code represents the actual code (written in IDL or in the programming language, e.g. C++ or Java) to be incorporated (e.g. as header file in C++) in the target code to implement the interface design
  - related to messaging or the data distribution mechanisms that implement the interface
  - coherent with middleware services defined in the Platform Specific Model and called by the software application to implement the interface
- Interface Design Descriptions and Interface Source Code are generated by the software development team that in charge of the software application that provides the interface (interface provider)
  - Interface Design Descriptions and Interface Source Code are used by all the software development teams that are in charge of software applications that require the interface (interface consumer)

Support to Systems Of Systems Integration and Testing

- Integration and Testing of Large Systems is a very complex task due the number and variety of interfaces
- Integration and Testing require Data Recording and Analysis tools to support the verification and validation of interfaces
  - tools able to parse, interpret and post-process, during the test analysis session, the data recorded during the test execution session
- Data Recording and Analysis tools need the knowledge of the interface design
  - in order to automate the data analysis it is necessary that the tools are aware of the semantic content of recorded data
- Data Recording and Analysis tools can acquire the knowledge of the interface design from the system architecture model
  - package containing the interface design can be exported from the model in XML files through XMI standard.
  - testing and integration tools can parse exported files to extract the interface definition catalogue used in the data analysis phase
  - this approach ensures the flexibility and adaptability of the testing and integration tools with respect to the interface design and system modelling tool

Issues for Configuration and Deployment of Systems Of Systems

- Configuration/Deployment of Systems Of Systems is a complex task
  - a System Of Systems could be distributed on different geographic areas, composed by hundreds of components, running under multiple HW constraints and on different resources
  - the configuration of a System Of Systems involves the production of many configuration files describing the structure of the System Of Systems in general, the configuration parameters of each component and how each component has to interact with the others.
- Issues
  - Complexity: configure properly the QOS of provided features
  - Coherence: configure properly all the system services
  - Comparison: find similarity among versions

Model Driven Configuration/Deployment of Large SW Systems

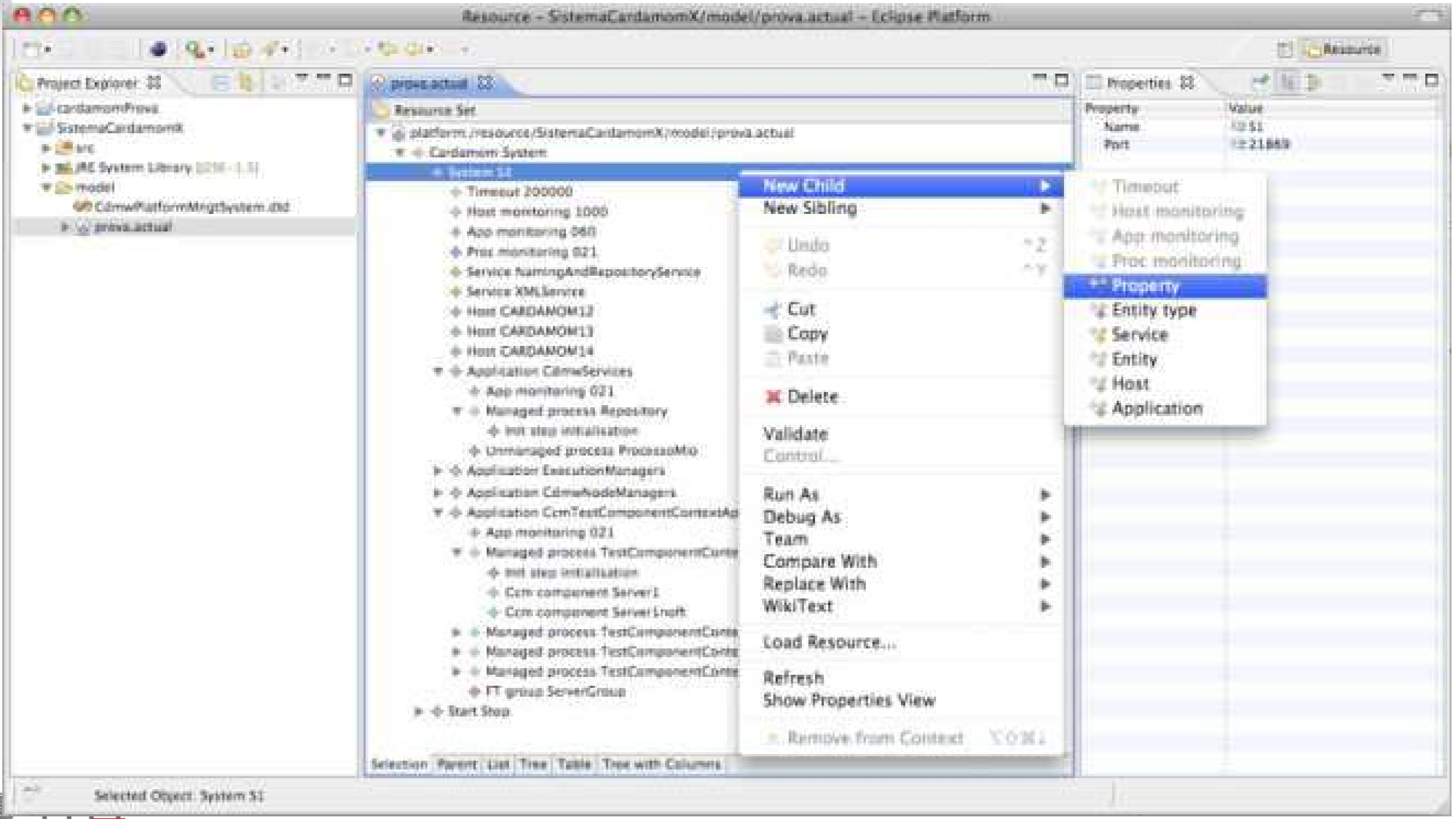
- Configuration and deployment of Systems Of Systems take benefits from Model Driven approach
  - Data modelled at PIM level can be extracted into XML (eXtensible Markup Language) or XMI files (eXtensible markup language Metadata Interchange) to produce files enabling data information exchange
  - It is possible to transform the data information files applying suitable stylesheets, written in XSL (eXtensible Stylesheet Language), to pick up the information of interest and complement the information with additional data, if necessary
  - XSL stylesheets can produce configuration files in the chosen format, normally XML, to be used at compilation time or run time to characterize the behaviour of the software applications
- Advantages of generating configuration files from the model
  - ensuring consistency of configuration files with system design information
  - allowing uniform and optimised style of configuration files across the whole system implementation
  - minimising the occurrence of human errors

Model Driven solution for System Configuration and Deployment

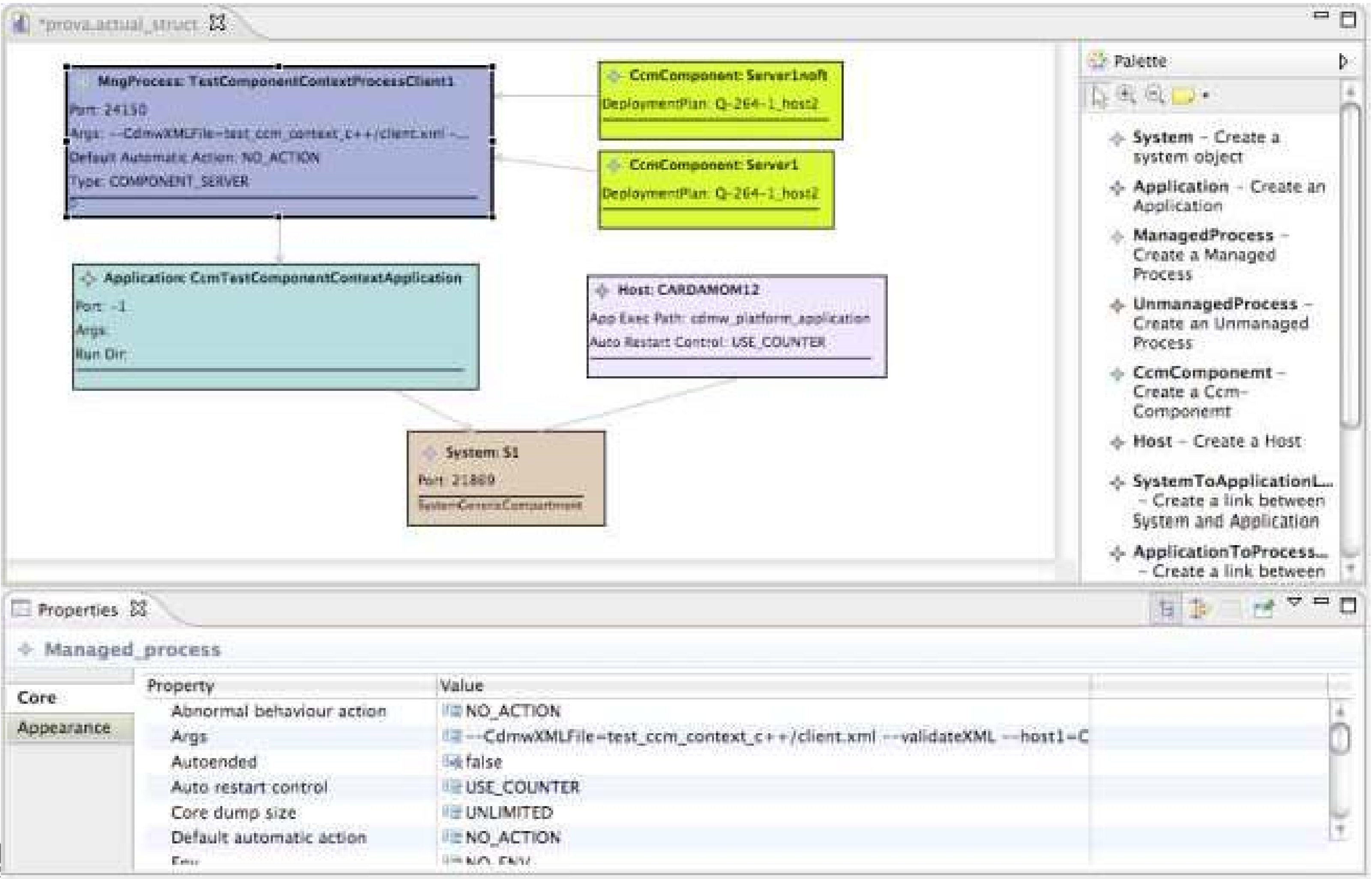
- Model Driven solution to configure and deploy Large SW Systems
  - UML-like HMI
  - Set Middleware concept on Graphical representation
  - Multiple view approach
    - Structure: how the system is composed
    - Deployment: how the system is distributed across physical environment
    - Activity: how the system services are interrelated for starting and stopping

SELEX Sistemi Integrati

Model Editor HMI



Snapshot of Structure View



*prova_actual_struct

MngProcess: TestComponentContextProcessClient1
Port: 24150
Args: --CdmwXMLFile=test_ccm_context_c++/client.xml --...
Default Automatic Action: NO_ACTION
Type: COMPONENT_SERVER

CcmComponent: Server1noft
DeploymentPlan: Q-264-1_host2

CcmComponent: Server1
DeploymentPlan: Q-264-1_host2

Application: CcmTestComponentContextApplication
Port: -1
Args:
Run Dir:

Host: CARDAMOM12
App Exec Path: cdmw_platform_application
Auto Restart Control: USE_COUNTER

System: S1
Port: 21869
SystemGenericCompartment

Palette

- **System** – Create a system object
- **Application** – Create an Application
- **ManagedProcess** – Create a Managed Process
- **UnmanagedProcess** – Create an Unmanaged Process
- **CcmComponemt** – Create a Ccm-Componemt
- **Host** – Create a Host
- **SystemToApplicationL...** – Create a link between System and Application
- **ApplicationToProcess...** – Create a link between

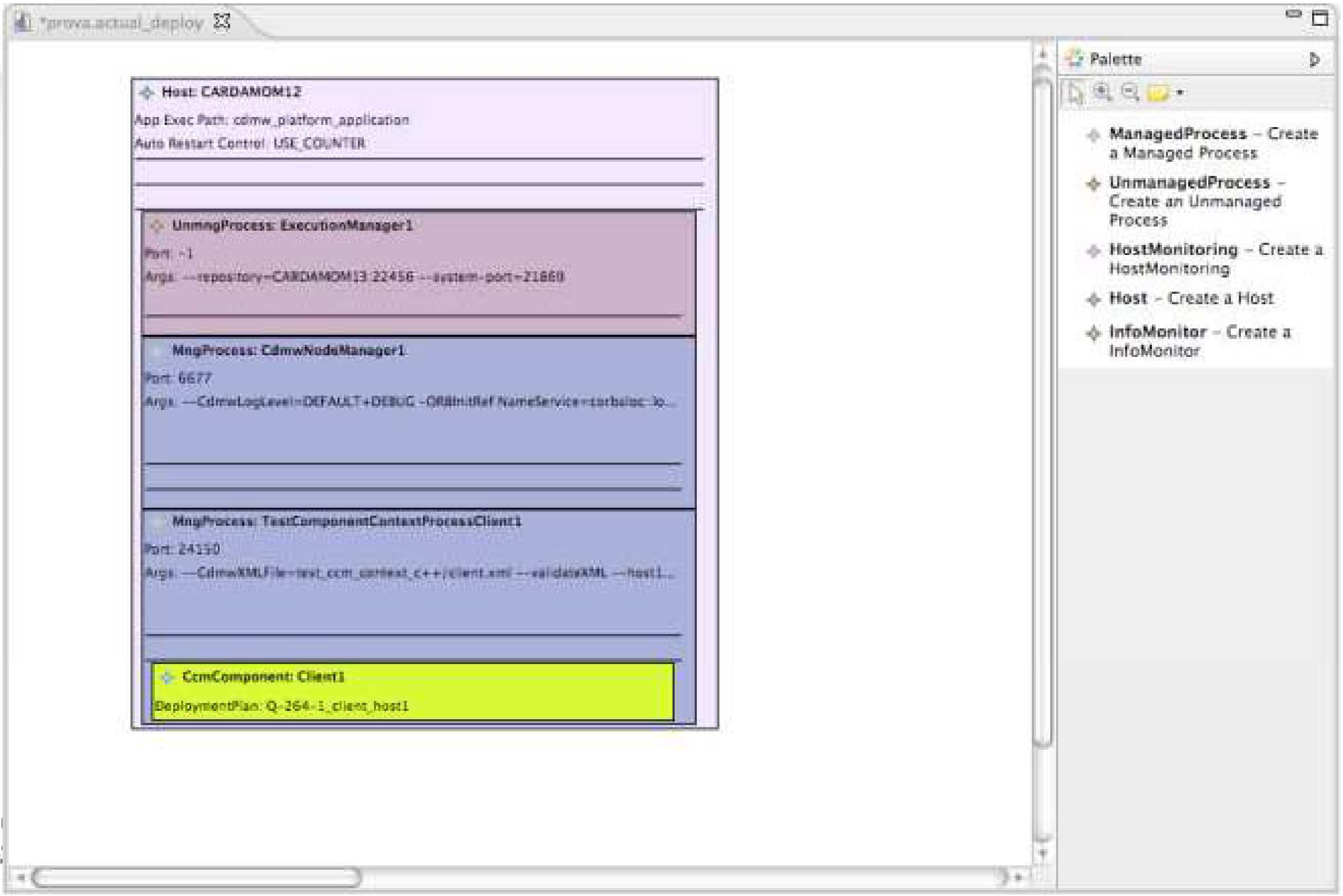
Properties

Managed_process

Core
Appearance

Property	Value
Abnormal behaviour action	NO_ACTION
Args	--CdmwXMLFile=test_ccm_context_c++/client.xml --validateXML --host1=C
Autoended	false
Auto restart control	USE_COUNTER
Core dump size	UNLIMITED
Default automatic action	NO_ACTION
Env	NO_ENV

Snapshot of Deployment View



Host: CARDAMOM12
App Exec Path: cdmw_platform_application
Auto Restart Control: USE_COUNTER

UnmngProcess: ExecutionManager1
Port: -1
Args: --repository=CARDAMOM13 22456 --system-port=21869

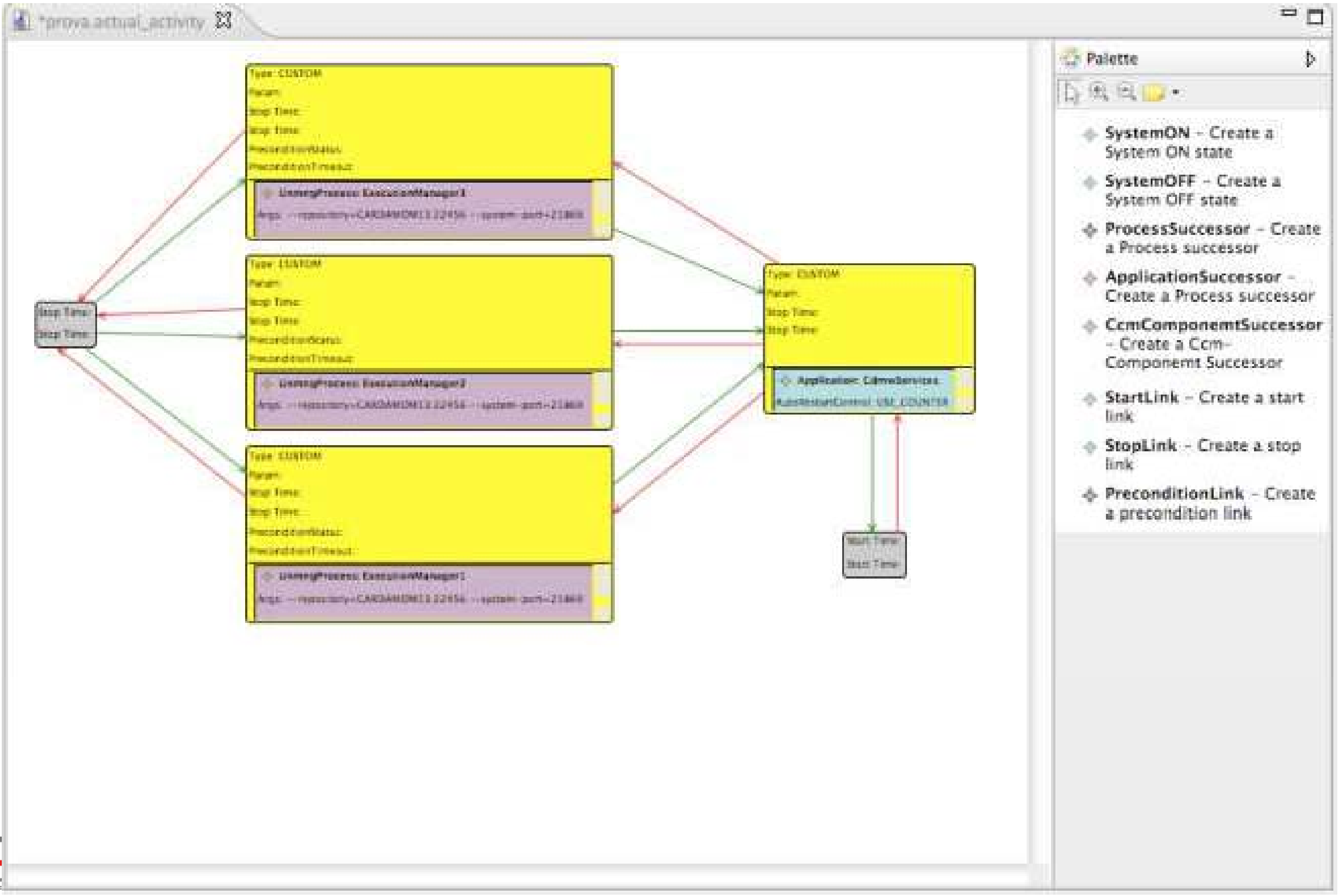
MngProcess: CdmwNodeManager1
Port: 6677
Args: --CdmwLogLevel=DEFAULT+DEBUG -ORBInitRef NameService=corbaloc::lo...

MngProcess: TestComponentContextProcessClient1
Port: 24150
Args: --CdmwXMLFile=test_ccm_context_c++/client.xml --validateXML --host1...

CcmComponent: Client1
DeploymentPlan: Q-264-1_client_host1

ManagedProcess – Create a Managed Process
UnmanagedProcess – Create an Unmanaged Process
HostMonitoring – Create a HostMonitoring
Host – Create a Host
InfoMonitor – Create a InfoMonitor

Snapshot of Activity View


*prova.actual_activity
Type: CUSTOM
Param:
Start Time:
Stop Time:
PreconditionStatus:
PreconditionTimeout:
Stop Time:
Stop Time:
Start Time:
Stop Time:
Palette
SystemON - Create a System ON state
SystemOFF - Create a System OFF state
ProcessSuccessor - Create a Process successor
ApplicationSuccessor - Create a Process successor
CcmComponemtSuccessor - Create a Ccm-Componemt Successor
StartLink - Create a start link
StopLink - Create a stop link
PreconditionLink - Create a precondition link

SELEX Sistemi Integrati S.p.A.

Via Tiburtina Km 12.400
00131 Rome, Italy
+39 06 41501 – info@selex-si.com