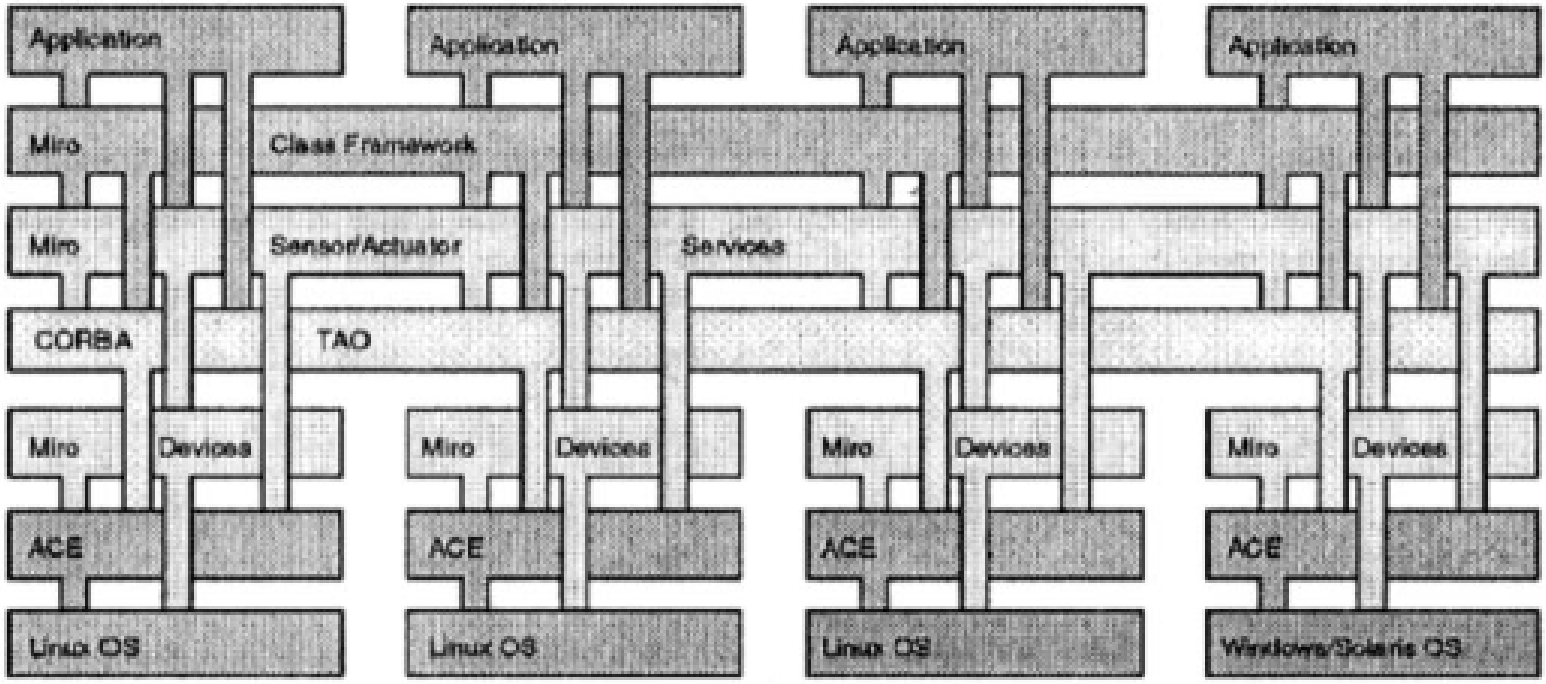
CpSc 875

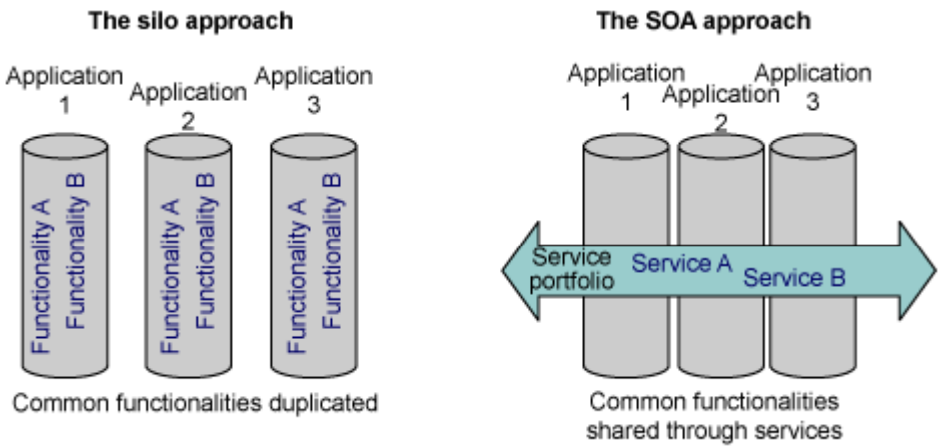
John D. McGregor

C11 - Documentation

Blended Architecture

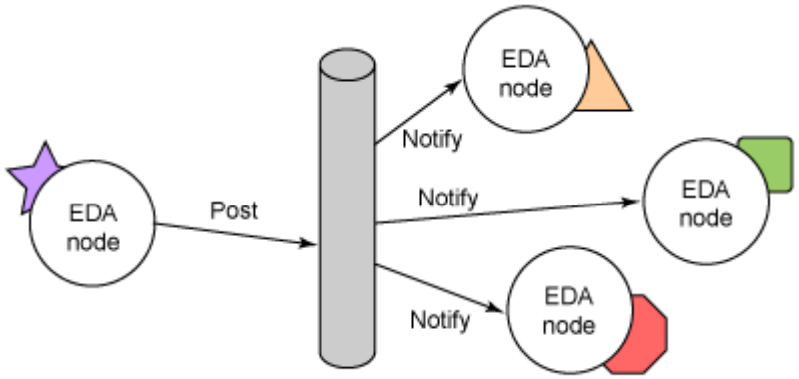


Common services

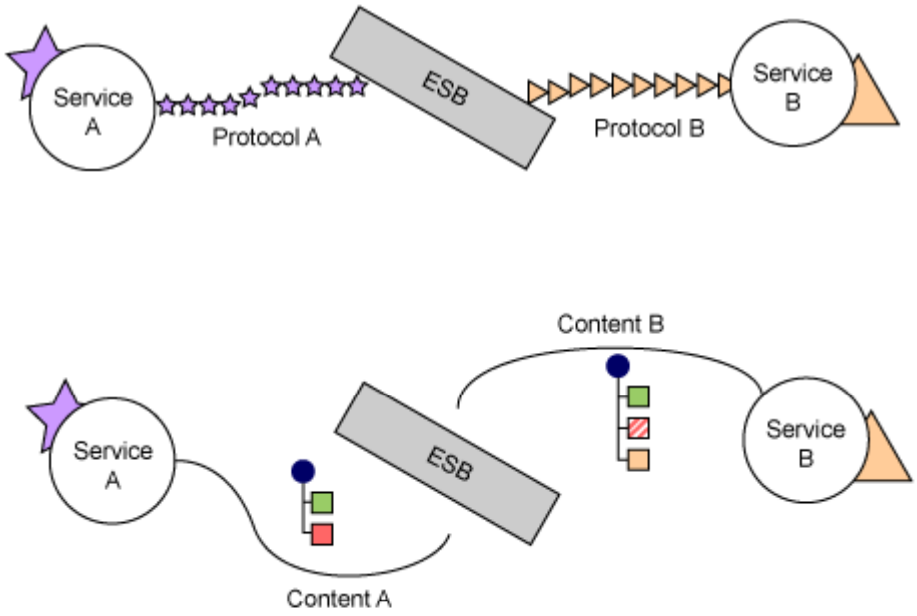


Request mechanism for service oriented architecture is variable

Event architecture



Mediation by the bus



Services

Transport services must ensure the delivery of messages among the business processes interconnected via the enterprise bus. Transport also includes content-based routing. It means it can direct messages to different destinations. As part of a mission-critical environment, these services are transactional, secured and monitored.

Event services provide event detection, triggering and distribution capabilities. They are related to the notion of event processing, a technique for analyzing and controlling the complex series of interrelated events in Event-Driven Architectures (EDA). Event-driven paradigms are not new. However, they are gaining industry momentum and represent the core concepts of emerging technologies like Complex Event Processing (see Resources).

Mediation services address two different purposes. First, the mediation ensures the necessary protocol matching to integrate heterogeneous systems. As two different services do not have to use the same transport protocol, the mediation service takes care of the transformation from one protocol to the other, so that the communication is possible. The protocol switch is transparent for all the participating services of a business transaction.

2 sources

- Clements et al. – book that describes an approach called Views and Beyond
- IEEE 1471 adopted standard

- Results in a two volume set of documentation. See the reference at the end of the slides to the pedagogical product line and follow the link to see an example two volume set.

Views and Viewpoints

- 1. Module views describe how the system is to be structured as a set of code units.
- 2. Component-and-connector (C&C) views describe how the system is to be structured as a set of interacting runtime elements.
- 3. Allocation views describe how the system relates to non-software structures in its environment.

Viewpoint

- Presents the information in the architecture from a certain perspective
- That is, it emphasizes certain aspects of the architecture over other aspects
- Example, the specification viewpoint shows the spec without showing the implementations available; this communicates more with the designer than the programmer

Viewpoint

- define the types of elements, the relations among them, the significant properties they exhibit, and the constraints they obey for views conforming to this viewpoint.
- The basic structures in an architecture are one source of viewpoints
- Module viewpoint – Shows an individual module and information about that module

View

- Applies a viewpoint to the architecture for a specific stakeholder
- Every stakeholder should have at least one view that speaks to their concerns
- The view should put the information into context but not confuse the reader with extraneous material

View

- For example, a user interface designer would want a view that shows each module used in the interface from the viewpoint of that module's definition.

View

- Name of view
- View description
- Primary presentation
- List of view packets
- Element catalog
- Context diagram
- Variability mechanisms
- Architecture background

View Packet

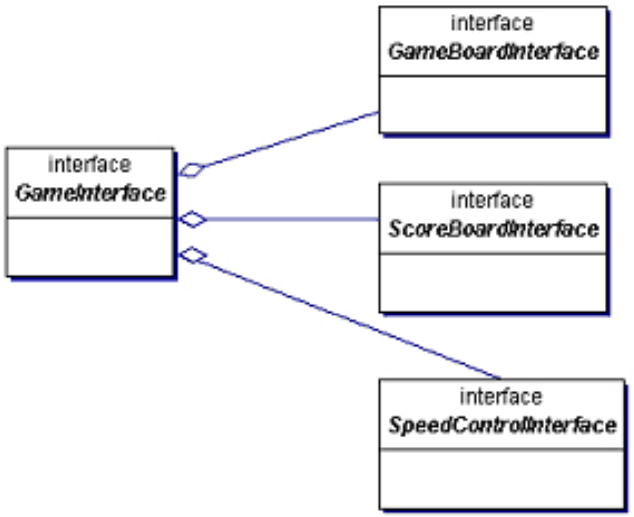
- Primary Presentation
- Element Catalog
- Properties of the elements
- Relations and their properties
- Element interfaces
- Element behavior
- Context Diagram
- Variation Guide
- Architecture Background

  Rationale.
  Analysis results.
  Assumptions.
- Other Information
- Related View Packets

Example

- **Module Decomposition View**
- In this section, we describe the basic structure of an AGM game. Game variations are addressed by substitution, parameterization, and specialization. Game-specific behaviors are provided by game-specific implementations of the interfaces in this document.
- **Module Decomposition View Packet 1: Game**
- The previous figure shows the Game component as the representation for the generic product. The following figure shows that component's interface as the top-level interface of the system (defined in the GameBoard Interface section) and the other major interfaces that are at the first level of decomposition within the GameInterface (defined later in this document).

- Primary presentation



Element Catalog

Elements and their properties. The following table displays element details.

Relations and their properties. The primary relation is composition. Game is responsible for creating, managing, and killing the elements it composes.

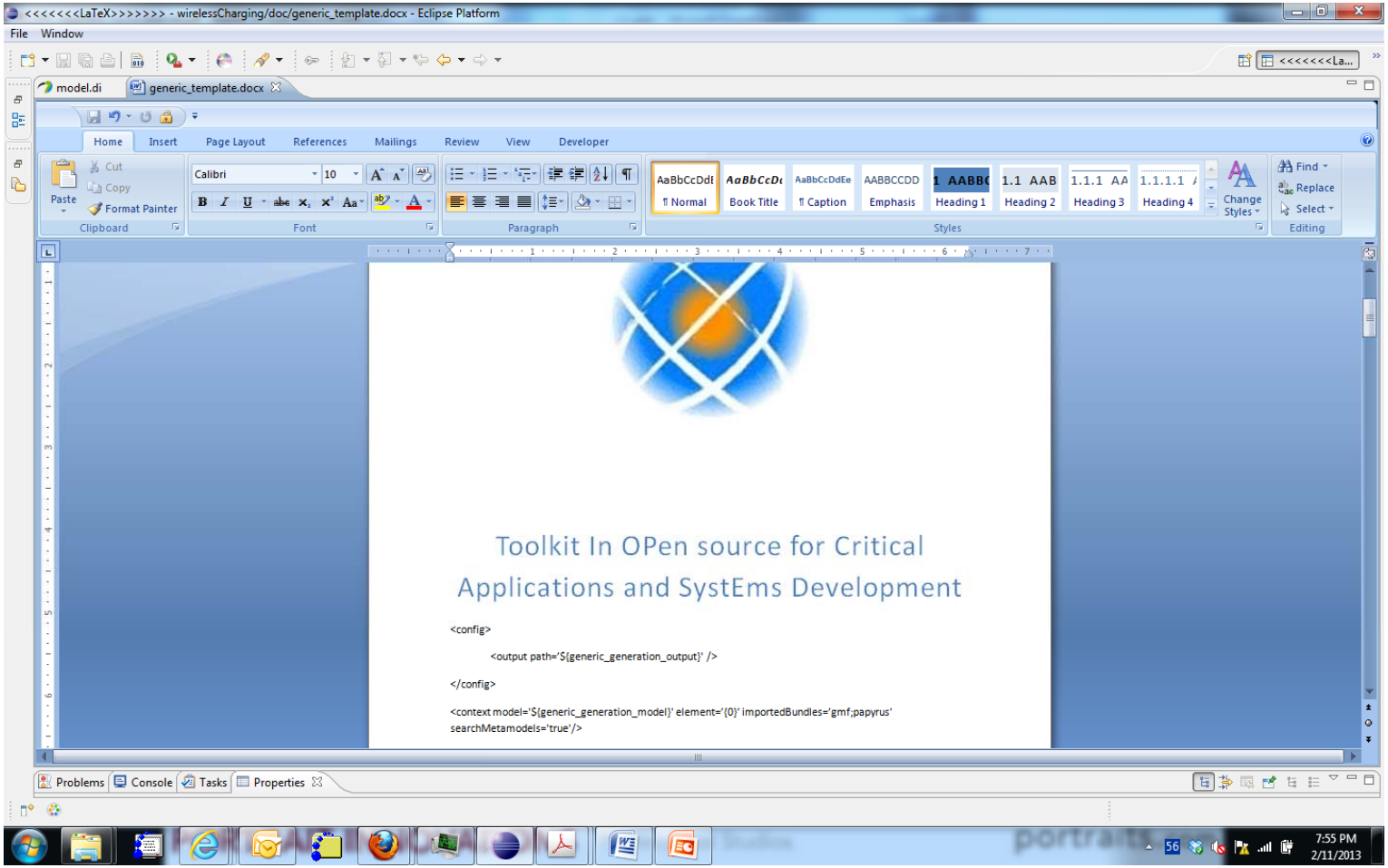
Elements and Responsibilities for Model Decomposition View Packet 1: Game

Element	Responsibilities
GameBoard interface	This container component holds all the elements needed for the game.
ScoreBoard interface	This interface keeps and presents the score as the game specifies. It is defined in the ScoreBoard Interface section.
SpeedControl interface	This interface controls how often a tick is issued to the GameBoard. It is defined in the SpeedControl Interface section.

- **Relations and their properties.** The primary relation is composition. Game is responsible for creating, managing, and killing the elements it composes.
- **Element interfaces.** Game interface is the program's GUI. It provides the user with Game start, stop, pause, and save behaviors. The other interfaces are documented as follows:
- GameBoard
- SpeedControl
- ScoreBoard
- **Element behavior.** The behavior is the game that is displayed to the user.
- Context Diagram
  Game is the top level of the product and the top-level context.
- Variation Guide
  Game and ScoreBoard will each be replaced by a game-specific implementation as described in the Module Generalization View section.

- Variation Guide
  Game and ScoreBoard will each be replaced by a game-specific implementation as described in the Module Generalization View section.
- Architecture Background
  Game encapsulates game-specific behavior. It arranges the GameBoard, keeps score, and determines the won/lost status based on its rules. The composed elements are mostly generic and can be reused and rearranged to implement other games.
- Other Information
  No other information applies.
- Related View Packets
- Module Generalization View Packet 1: Game
- ScoreBoard Interface

GenDoc2 Template



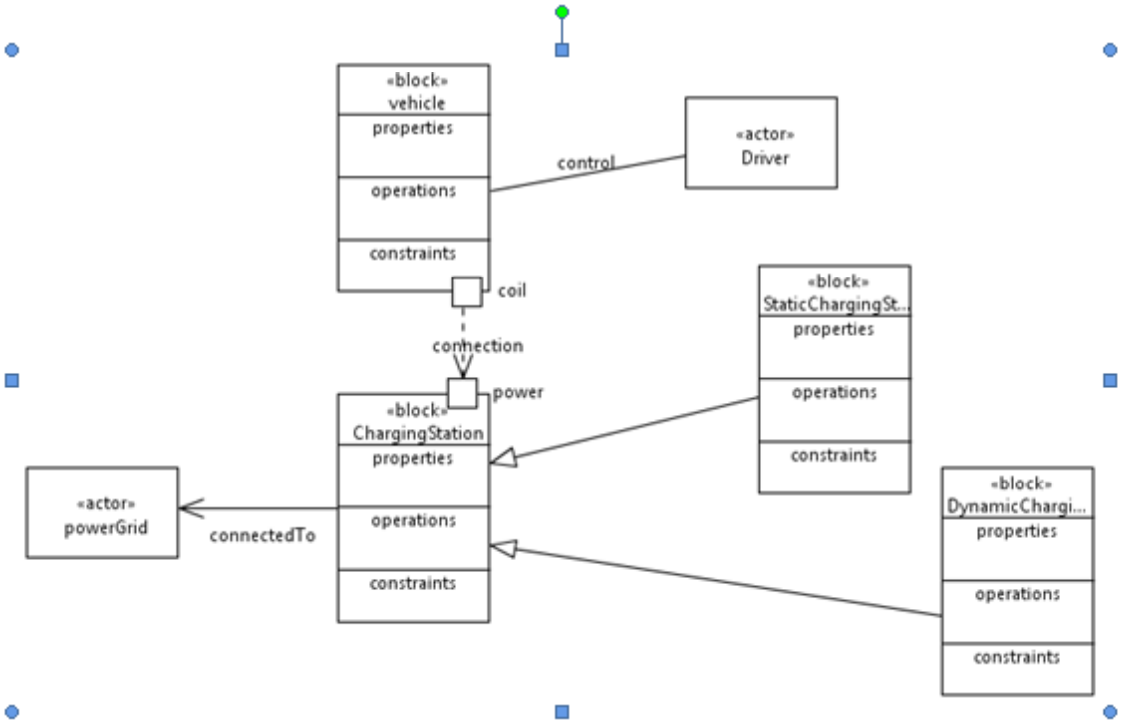
Tutorial for GenDoc2 at www.topcased.org – works on Topcased not OSATE

GenDoc2 fragment

```
<gendoc>
[for (d : notation::Diagram | di::PageRef.allInstances().emfPageIdentifier->asOrderedSet())]
    <drop/>
DIAGRAM : [D.NAME/]
<image object='[d.getDiagram()/]' maxW='true' keepH='false'>
</image>
[for (e: ecore::EObject | getElementsInDiagram(d))]<drop/>
[if(e.getDocumentation().size() > 0)]<drop/>
[CLEAN(GETTEXT(E))/]
[clean(e.getDocumentation())/]

[/if]<drop/>
[/for]<drop/>
[/for]<drop/>
</gendoc>
```

GenDoc2



3 DIAGRAM : TOYOTA

Next steps

- Read http://www.sei.cmu.edu/library/abstracts/reports/05tn017.cfm
- Create the first draft of the documentation for your architecture.
- Select the “download the template” link on:
  http://www.sei.cmu.edu/architecture/tools/viewsandbeyond/
- Here is example documentation for the pedagogical product line architecture: http://www.sei.cmu.edu/productlines/ppl/arch_docs.html
- Fill in the template for your architecture; add the GenDoc2 control info to your template and generate the SysML diagrams for parts of the documentation. Add OSATE diagrams as screen prints.
- Apply the “Check Latency Analysis” tool and include the results from that in the package.
- Submit by Feb 18; 11:59pm