Component Models

Definitions:

A ***Component*****,** for this discussion only, is a relatively independent part of an IT System and is characterized by its responsibilities, and the interfaces it offers.

A *Component Model* describes the hierarchy of functional components, their responsibilities, static relationships, and the way components collaborate to deliver required functionality

Description. A Component Model evolves through several stages, taking into account the successive allocation of system functions to actual systems (components) and the relative maturity of the architecture description's Systems Viewpoint (SV) as a whole. As the process of developing the architectural description advances, the use of specific models is decided upon, middleware is chosen, variant technologies are tried and approved; all such decisions end up being reflected in the final Component Model.

The first, level of elaboration -- *Conceptual* -- describes the architecture at a "macro" level and aligns its representation in accordance with generally-accepted representational principles recognized by the architecture discipline. Conceptual elaboration emphasizes increasing cohesion within layers of the architectural description and reducing coupling that may exist between them, thus enabling the component to be re-used by another organization. This level of elaboration is meant to be technology-agnostic – that is, it can be implemented using any toolset that supports the discipline principles.

The second level of elaboration (*Specification*) helps to further structure and refine the architectural representation by adding technology elements – such as (in the case of a software architecture) transport mechanisms, programming models, and protocols. Like the Conceptual level, the Specification level is tool-agnostic. This means that it can be implemented using any technology that allows the selected protocols and programming model to be easily included in the architectural representation.

The third level of elaboration (Physical) *realizes* the logical components which were identified at the Specification level. It selects from among the set of all implementation alternatives (e.g., hardware and software packages, technologies) the ones that *will be employed* in the final system architecture. The Physical model may be closely tied to, and depend upon, the application development tool the builder uses to actually implement the system.

Construction. A Component Model is described using three views:

1. Component Relationship Diagram
2. Component Description
3. Component Interaction Diagram

Each of technique provides important information about the Component Model. The Component Relationship Diagram and Component Descriptions provide a *static* view of the model. The Component Interaction Diagram provides a *dynamic* view – that is, insight into how the model's various components interact when the system responds to an event, request, etc.

Depict Component Relationships

Component Relationship Diagrams can be created using a UML Class Diagram. **Figure 1** depicts such a Diagram. Note: use of UML notation here does not imply that all components must be coded in an object-oriented language.

An initial high-level component model diagram can be created quickly to show the overall topology of major functional aspects of the system. This view is not yet detailed enough to understand fully what each package will contain, but does allow stakeholders to understand the major features and evaluate the completeness of the architectural description.

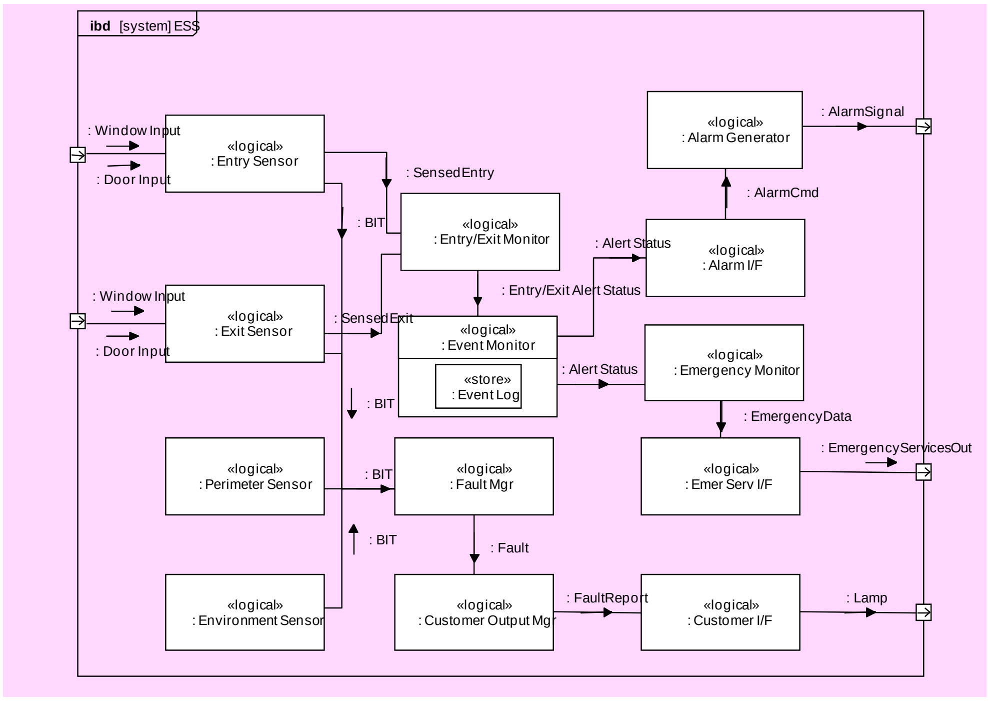

Figure 1: Component Relationship Diagram

Create Packages

The next step is to add more detail to the Component Relationship Diagram, to create a second iteration called a *Package Diagram*. **Figure 2** depicts such a Diagram. The Package Diagram is used to organize the various components in the Component Model; it does so by grouping them together in a name space. In the Package Diagram shown below, packages from the first diagram are shown in more detail in order to clarify the actions each of the original (high-level) packages is intended to support.

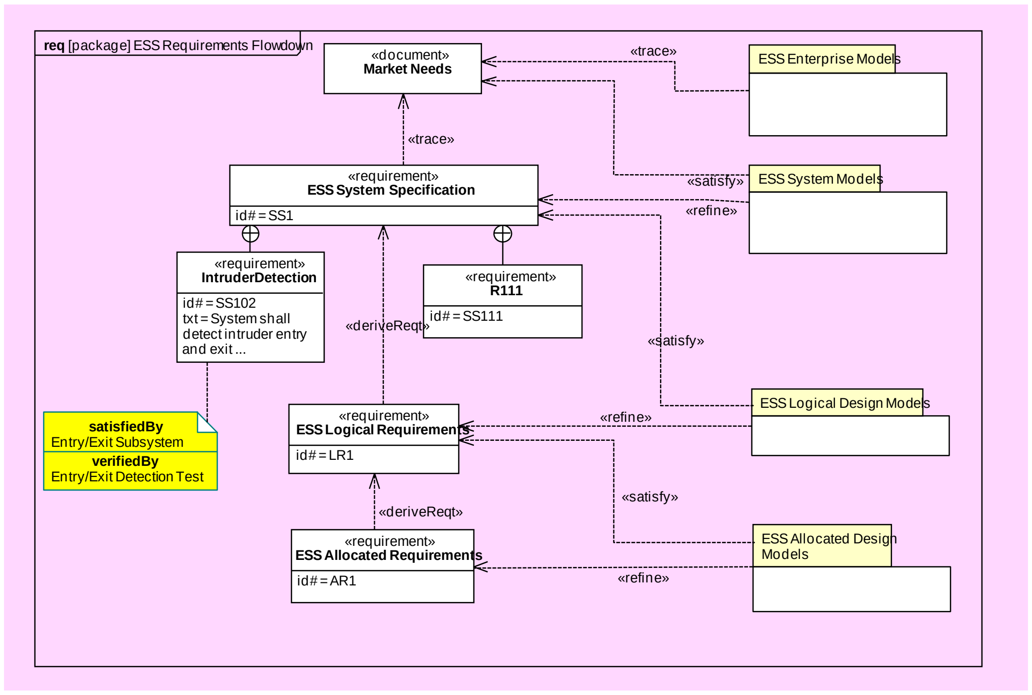
req [package] ESS Requirements Flowdown
«document» Market Needs
«trace»
ESS Enterprise Models
«trace»
«satisfy»
ESS System Models
«refine»
«requirement» ESS System Specification
id# = SS1
«requirement» IntruderDetection
id# = SS102
txt = System shall detect intruder entry and exit ...
«requirement» R111
id# = SS111
«deriveReqt»
«satisfy»
satisfiedBy
Entry/Exit Subsystem
verifiedBy
Entry/Exit Detection Test
«requirement» ESS Logical Requirements
id# = LR1
«refine»
ESS Logical Design Models
«satisfy»
«deriveReqt»
«requirement» ESS Allocated Requirements
id# = AR1
ESS Allocated Design Models
«refine»

Figure 2: Package Relationship Diagram

Depict Component Interactions

All models should provide both static and dynamic descriptions. A model should not be considered complete without both descriptions. The dynamic description of the Component Model can be represented in a Component Interaction Diagram ("Activity Diagram").

A Component Interaction Diagram describes a particular collaboration between components – i.e., a possible runtime execution. Component Interaction Diagrams show how the services requested from a component are realized through collaborations among its contained components. Exchanges that occur between any two components during collaboration are called "interactions." A Component Interaction Diagram showing collaborations between the top-level components describe *system-wide* interactions.

Tip: Appropriate Levels of Detail. Each component in the Component Model needs to be described to a level of detail that is directly related to the level of elaboration of its containing model. The amount of detail needed in a component description is closely related to *who will use* the model and *for what purpose(s)* they will use it. The Conceptual and Specification level models are used primarily by the architect as steps toward a Physical level model. The Physical level model is typically used as input to fine-grained design activities, at which point there is often a hand-off from architecture to engineering. The Physical level of elaboration can be thought of as the "payoff" level for system developers, since based on a Physical representation of the architecture an implementation contractor (builder) can proceed to produce a detailed system design. This implies that the Physical level model needs to describe components to the level of detail needed by a designer.

Conceptual-level component descriptions are typically brief and succinct. At the Specification level, decomposition of the model into more fine-grained components means that additional detail is needed to clarify the roles of the components. At the Physical level of elaboration, greater detail should be supplied so that there is no chance for miscommunication between architect and engineer. For projects where the sponsoring organization plans to implement the initial capability using off-the-shelf applications with minimal customization, the Physical level of elaboration should describe existing COTS/GOTS capabilities as well as the estimated amount of integration needed to make them work in the existing infrastructure environment.