REFERENCES


BIBLIOGRAPHY


Bibliography


GLOSSARY

activity
Task(s) of the project that consumes time while people/equipment either work or wait.

activity duration
Estimate of time (hours, days, weeks, months) necessary to complete a project task.

application
Software designed to fulfill the customer’s product requirements.

architectural pattern/style
Define element types and how they interact.

binding
The binding relation is used when a conceptual component or connector is decomposed, either to bind an inner port to a port of the enclosing component or to bind an inner role with a role of the enclosing connector.

central design task
The process of defining the elements of an architecture view and the relationships among them and defining how these elements are configured. Strategies developed during global analysis are used to guide design decisions and to improve the ability of the system to respond to change.

code architecture view
The code architecture view is a view of the software architecture. For this view, the architect determines how runtime entities from the execution view are mapped to source components and how the deployment components are produced from the source components.

communication path
One of the basic elements in the execution view. The communication paths identify the expected allowable communication between runtime entities, including the mechanisms and resources used for the communication. Defining communication paths is a central design task in the execution view.

**component**

The conceptual components or code architecture elements. Note that UML defines a component as a modular, reusable unit of implementation. This corresponds to components in the code architecture view.

**conceptual component**

Conceptual components are independently executing peers within a component-connector model. A critical goal of this model is to keep the control aspects of the components simple, concentrating functional behavior in the components and control aspects in the connectors. A component interacts with its surroundings through ports. Defining conceptual components is a central design task in the conceptual view.

**conceptual configuration**

The relations among components and connectors. A conceptual configuration that contains component and connector types constraints how instances of these types can be interconnected. A conceptual configuration that contains instances defines which instances exist in the product and how they interconnect. Defining conceptual configuration is a central design task in the conceptual view.

**conceptual connector**
Connectors mediate the interactions among the conceptual components. A connector’s roles define the behavior of the participants in the interaction. Defining conceptual connectors in a central design task in the conceptual view.

**conceptual view**

The conceptual view is a view of the software architecture. In this view, the functionality of the system is mapped to architecture elements called conceptual components, with coordination and data exchange handled by elements called connectors.

**connection**

Components and connectors are interconnected through their ports and roles. Connections are possible only when the associated protocols are compatible and only when the elements are nested within the same component and connector.

**connector**

See conceptual connector.

**Critical Path Method (CPM)**

A scheduling method based on the estimates of time required to complete activities on the critical path. The method computes early, late and slack times for each activity in the network. It establishes a planned project duration, if one is not imposed on the project.

**decomposition**

The decomposition of a system captures the way the system is locally decomposed into subsystems and modules.

**deployment component**

Deployment components are required for the system to start up, to operate, to be used and to be shut down. The deployment code components include executables, shared or dynamic
libraries, the system configuration files and resources such as storage for data and shared memory. Defining deployment components is a central design task in the code view.

**design pattern**
A recurring design problem with its generic solution. Design patterns are a technology that may be relevant to architecture or may be relevant only to detailed design. When a design pattern describes the interactions of architecture elements, it is considered to be part of the software architecture. When it describes the structure and interactions within architecture elements, it is part of the detailed design.

**domain**
An area of knowledge characterized by a set of concepts understood by practitioners of that area; often used as application domain to denote the area an application addresses.

**duration**
The time needed to complete an activity, a path or a project.

**event**
A point in time when an activity(s) is started or completed. It does not consume time. It also refer to an action or occurrence detected by a program. Events can be user actions or system occurrence.

**execution configuration**
Describes the system’s runtime topology by characterizing the instances of the runtime entities and how they interconnected. Defining the execution configuration is a central design task of the execution view.

**execution view**
The execution view is a view of the software architecture. It describes how modules are mapped to the elements provided by the runtime platform and how these are mapped to the hardware architecture. The execution view defines the system's runtime entities and their attributes, such as memory usage and hardware assignment.

**framework**

Frameworks specify structure and the flow of control and/or data. The specification is in the form of a partial implementation – an implementation of infrastructure.

**global evaluation**

The processes of continual evaluation of design decisions to see the results of the global analysis are being followed. During the central design tasks, interactions among decisions must be evaluated periodically.

**interface**

The services of a module are defined by the interfaces it provides. A module may also need the services of another module to perform its function. These services are defined by the interfaces it requires. This is an approximate but not exact fit with UML semantics.

**intermediate component**

Intermediate components are specific to the implementation language and the deployment tools. For C++, each .CPP file gives rise to an .obj file. The .obj components are related to their respective source components through the dependency relation. Defining intermediate components is a central design task in the code view.

**layers**
Layers organize the modules into a partially ordered hierarchy. A module is assigned to a layer, which then constraints its dependencies on other modules. Defining layers is a central design task in the module view.

**meta-model**

Describes how the elements of a view are related to each other. This is in contrast to a normal UML model, which we reserve for describing the architecture elements of a particular system. There is nothing in the notation that distinguishes a model from a meta-model, but the term meta-model tells you that you’ve stepped up a level of abstraction.

**milestone**

An event that represents significant, identifiable accomplishment toward the project’s completion.

**module**

A module encapsulates data and operations to provide a service. These services are defined by the interfaces it provides. A module may also need the services of another module to perform its function; these services are defined by the interfaces it requires. Defining modules is a central design task in the module view.

**module view**

The module view is a view of the software architecture. In this view, the components and connectors from the conceptual view are mapped to subsystems and modules. Here the architect addresses how the conceptual solution can be realized with today’s software platforms and technologies.

**network**
A logic diagram arranged in a prescribed format consisting of activities, sequences, interrelationships and dependencies.

**objective**
An end you seek to create or acquire. Should be specific, measurable, realistic, assignable and include a time frame for accomplishment.

**organizational factors**
Organizational factors arise from the business organization. Organizational factors constrain the design choices while the product is being designed and built. They are external to the product and exert a one-time, indirect influence on it. They include aspects of the project such as the schedule and budget and the skills and interests of the people involved.

**path**
A sequence of connected activities.

**platform element**
The element available in a hardware or software platform. In the execution view, a runtime entity is mapped to an element of the software platform.

**port**
The interaction points for the conceptual component. Ports define both incoming and outgoing messages so the components can truly be peers. Each port has an associated protocol that mandates how the incoming and outgoing operations are ordered.

**product line**
A family of similar products marketed by a company.

**project**
A complex, non-routine, one-time effort to create a product or service limited by time, budget and specification.

**project life cycle**

The stages found in all projects – definition, planning, execution and delivery.

**protocol**

A protocol describes how conceptual components and connectors coordinate and synchronize their interactions and communicate with each other. A protocol is defined as a set of incoming message types, outgoing message types and the valid message exchange sequences.

**reference architecture**

Defines elements types, allowed interactions and how the domain functionality is mapped to the architecture elements. Similar to domain-specific software architecture.

**resource**

Any person, groups, skill, equipment or material used to accomplish a task, work, package or activity.

**resource allocation**

The runtime instances and budgets defined in the execution configuration are allocated to particular hardware devices and are assigned specific values to the budgeted attributes. This is a final design decision task of the execution architecture view.

**resource budgeting**

Assigning resource budgets to the conceptual components and connectors in the conceptual configuration. These budgets are refined later during resource allocation in the execution view. The initial budgets help identify potential resource problems. Resources include
devices, memory and compute time from the operating system. This is a final design decision task of the conceptual architecture view.

risk
The chances that an undesirable project event will occur and the consequences of all its possible outcomes.

runtime entity
One of the fundamental elements in the execution view. The modules from the module view are assigned to runtime entities. A runtime entity is allocated to one of the platform elements defined for the software platform. There may also be runtime entities such as daemons or other server processes that have no direct correspondence to modules but are needed to support the other runtime entities. Defining runtime entities is a central design task in the execution view.

runtime platform
The hardware and software platform on which a system (product) runs.

slack
Time an activity can be delayed before it becomes critical.

software platform
The infrastructure software between the product and the hardware platform, which traditionally has been the operating system. Today it certainly includes the operating system, but there may also be networking software, other middleware layers on top of the operating system, or something like a database management system, which is considered to be part of the software platform. Often products within a company share a common custom software platform, particularly when they are part of a product line.
source component
Source components are language-specific interfaces and language-specific modules or components that produce them. For languages such as C and C++, language-specific modules and interfaces are files named *.h, *.H, *.c and *.CPP. For languages such as Ada, these components are package specification and package implementation. Defining source components is a central design task in the code view.

system architect
One who selects and configures the hardware of a system.

system architecture
Software architecture and the hardware environment. The system architecture shows the physical relationships among software and hardware components in the system.

task
See activity.

Unified Modeling Language (UML)
A generic-purpose modeling language that describes the static structure and dynamic behavior of a system. It is not a design method or a development process.

Work Breakdown Structure (WBS)
A hierarchy method that successively subdivides the work of the project into smaller detail.