Systems Architecture Model

Deployable Components

An information model is described here to tie together planning, specification, construction and deployment of large interrelated systems of systems typical of IT.   Tools based on this information model can have the effect of dramatically decreasing development costs and increasing the speed with which they can be built.
The centerpiece of the model is the Deployable Component; what many people would simply call and application. The creation of Deployable Components is a fitting centerpiece for the model because it represents the point at which planning and specification begin to yield value, and because the components are more easily understood by the end users and operational stakeholders.

Planning and Design Related Parts of the Model

Goals, Capabilities, Projects, Roadmaps, Business Information, System Groups, Systems

The planning related parts of the model are Goals, Capabilities, Projects and Roadmaps.
Business Goals (or just Goals) are units of organizational planning expressed in terms of things like customers, markets, products and services, etc., but generally described without any detail about the information involved in achieving them.
Capabilities are the things the business strategists believe they need to be able to do to achieve the Goals. For planning, *Capabilities should be expressed in terms of Business Information and Systems Groups*. System Groups are the collections of information functions that staff and industry people use to talk about what the business does. Business Information is simply the information that is needed to do business, and is specified in business conversational terms.
Projects are collections of capabilities that are anticipated to be created together as an executable unit of planning.

Roadmaps are ordered collections of Projects with an anticipated sequence, typically based on a balance of business value driven priority and some logical dependency order of building and extending Systems. Systems are subunits of systems groups that have narrower and more specific functions than System Groups.   Systems may be non-functional with respect to the business, whereas System Groups are used to describe the business and are infrequently non-functional. An important exception case arises when a technical system becomes a strategic part of the business, and some of its implementation details become part of a customer expectation, such as an IP address and port that a customer relies on for connectivity. When this happens, a piece of technical implementation has entered the business realm.

Tool Opportunity: Enterprise Roadmapping

Business Data Objects, Operations, Functions

A first tool opportunity based on this information model is to facilitate prioritization and project creation by making it easy to manage the prioritization of capabilities and their grouping into projects and to sequence those Projects int Roadmaps that achieve the greatest value per unit of time and expense.

A core function of system design is to describe first how capabilities can be divided up into Operations that Systems perform on Business Information, and then further describe the Business Data Objects and Functions performed by the Deployable Components that make up the Systems[1].

The cost of a Capability can be calculated if the technical solution is expressed in terms of System Operations (S. O.) on Business Information (BI) and there is an estimate for each S. O. on BI. The cost of a capability can alternately be calculated more precisely if the Deployable Component Functions (DC. F). on Business Data Objects (BDO). are known. In both cases, the cost of the capability is affected by the cost of dependent Capabilities and of Capabilities being created in the same project.

The S. O. on BI method lends itself to faster, but possibly less accurate estimates that can be used for prioritization planning. The DC. F. on BDO method could in theory be more accurate, and can be used for budgets for projects.

Tool Opportunity: Project Estimating

A second tool opportunity arises if DC. F. can be expressed in terms of patterns of known cost. The result would be more credible estimates, faster planning cycles, the ability to examine more planning possibilities, and more accurate budgets.

[1] For simple cases, or where there is more domain knowledge or stable existing systems it may be possible to skip directly to the specification of Deployable Component Functions on Business Data Objects.

Functional Specification Related Parts of the Model

Relationships, Participations, and Message Sets

The specification related parts of the model are Deployable Components (DC.) Relationships, Participations, and Message Sets.   The Functional specification of a technical solution will provide a Deployable Component Function explanation of how Business Data enters the Systems via some Function in the Operation category Create and moves through the Systems, including any Relationships and Message Sets that describe how data moves from DC to DC. Scaling and transport descriptions are left to be done separately using the Communication and Environment Parameter parts of the model.

The Relationships in the model are an abstraction of information exchanges amongst Deployable Components. Participations indicate the role of a Deployable Component in a Relationship as either a producer or consumer of messages in a Message Set. A Message Set elaborates on (joins to) a Relationship and carries information about the Business Data Objects and encoding shared by Deployable Components involved in the Relationship.

Tool Opportunity: Serializer Code Generation

Message specification systems represent a third tool opportunity, and in fact systems already exist that can generate working code to serialize and deserialize messages to and from memory structures in the run time deployments of components.

The Communication Parts of the Model

The specification of a technical solution can include Implementations, which describe how the Message is sent from one or more deployable components to one or more other deployable components.

The Environment Parameter Parts of The Model

A functional architecture can be set up with different deployment choices for various environments, and still be the same functional architecture from a business perspective. The environment parameter parts of the model dictate what parameters must have values to create a deployed environment from the architecture, and include Scale Parameters, Tech Stacks and Implementations for Relationships.

Tech Stack indicates the network or middleware protocol stack or file or other data conveyance implementation that may be used with the Implementation, Scale Parameter which indicates how additional deployed instances of Deployable Components should be configured to communicate to create larger system environments.

An extension of the code generation tooling for serializing and deserializing messages can used in component construction. Standard runnable containers suitable for the Deployable Components and libraries implementing the Tec Stacks used with the message code will form nearly complete applications, leaving only the function specifics to be implemented. Reducing the application construction work to focus on functions means that much of the remaining testing need can be done as unit testing, which is highly automated even ins existing standard practices. Structuring code generation in terms of data driven specification and standard libraries will greatly simplify what must still be communicated in order to efficiently orchestrate globally distributed development organizations.

Tool Opportunity: Environment Generation

A fourth tool opportunity, Environment Generation is to create a system that uses Scale Parameters, Tech Stacks, and deployment bundles to provision run containers, network names, and addresses to deploy a run environment for the architecture. [Is this of any use in a Zoo Keeper environment ?] This tool opportunity is a variation of the currently popular DevOps approach. The architecture specification driven approach can reduce the amount of custom scripting and overall time to automated deployments if there is sufficient libraries of runnable patterns that specification data instead of custom scripting is used to describe the runnable integrated System.

Service API Parts of The Model

The definition of service APIs should match the functional capabilities of components in the architecture, and are therefore mostly defined by the above mentioned model. Exposed Methods and Service Signatures further definen services. Services URLs take a standard form which can be dependant deployment names can have pre defined names provided for in deployment.

Additional tool opportunities:

- Operational monitoring related to business function.
- Security and compliance
- Change management
- … all IT functions.
- Visualization: functional
-