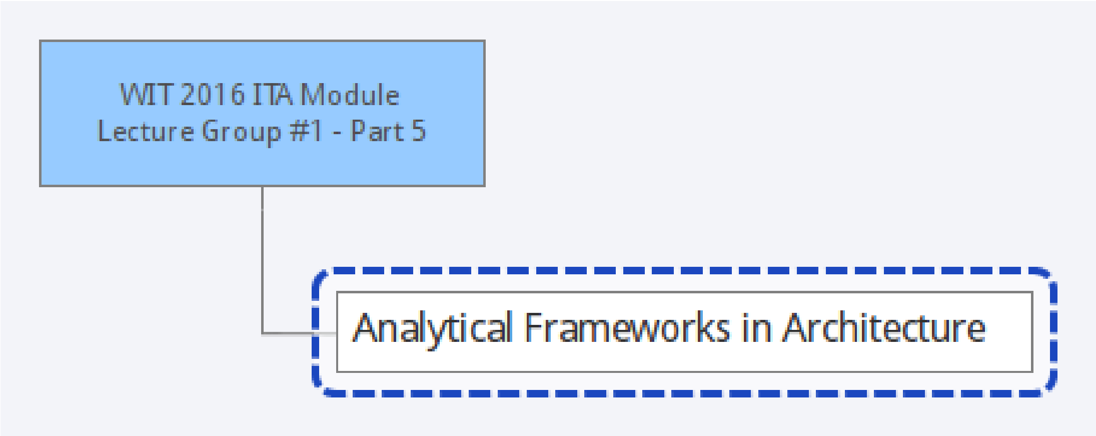
WIT 2016 ITA Module

Lecture Group #1 - Part 5
Analytical Frameworks in Architecture

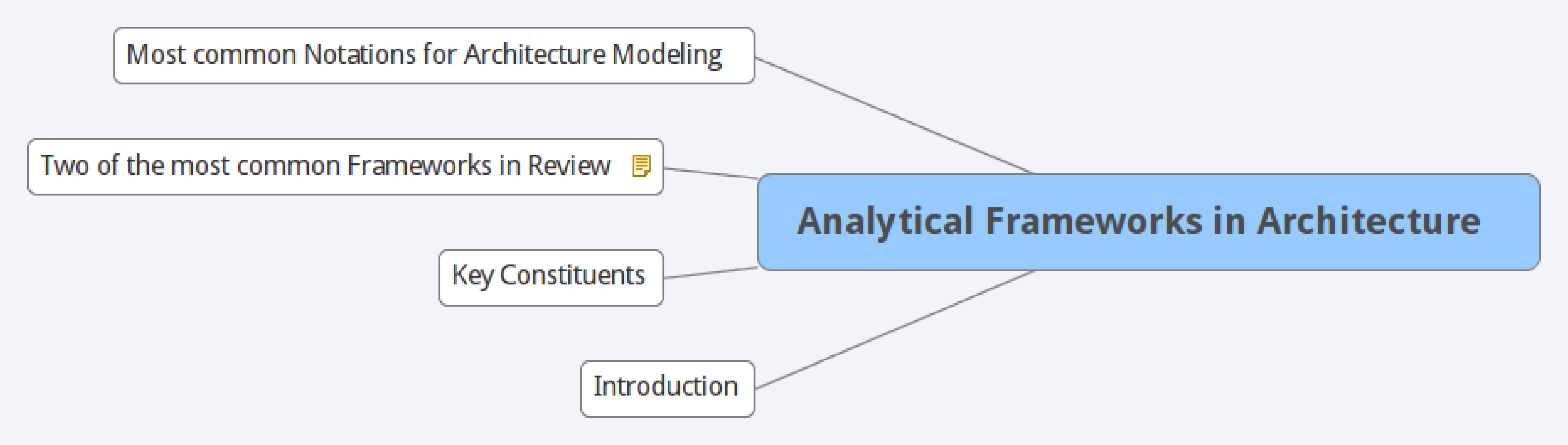
(cc) BY-NC-SA

Lecture Group #1 - Part 5


WIT 2016 ITA Module
Lecture Group #1 - Part 5
Analytical Frameworks in Architecture

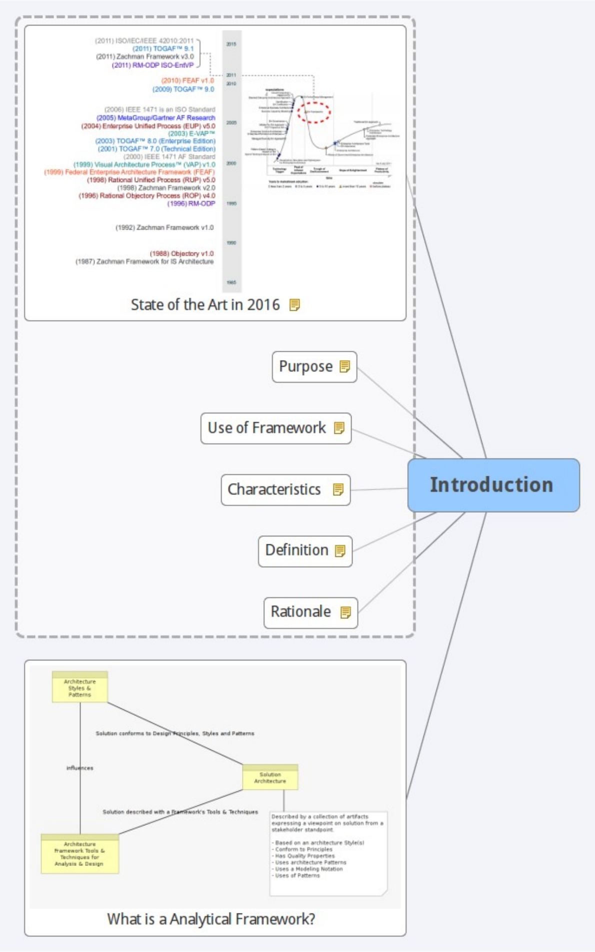
cc BY-NC-SA

Analytical Frameworks in Architecture


Most common Notations for Architecture Modeling
Two of the most common Frameworks in Review
Analytical Frameworks in Architecture
Key Constituents
Introduction

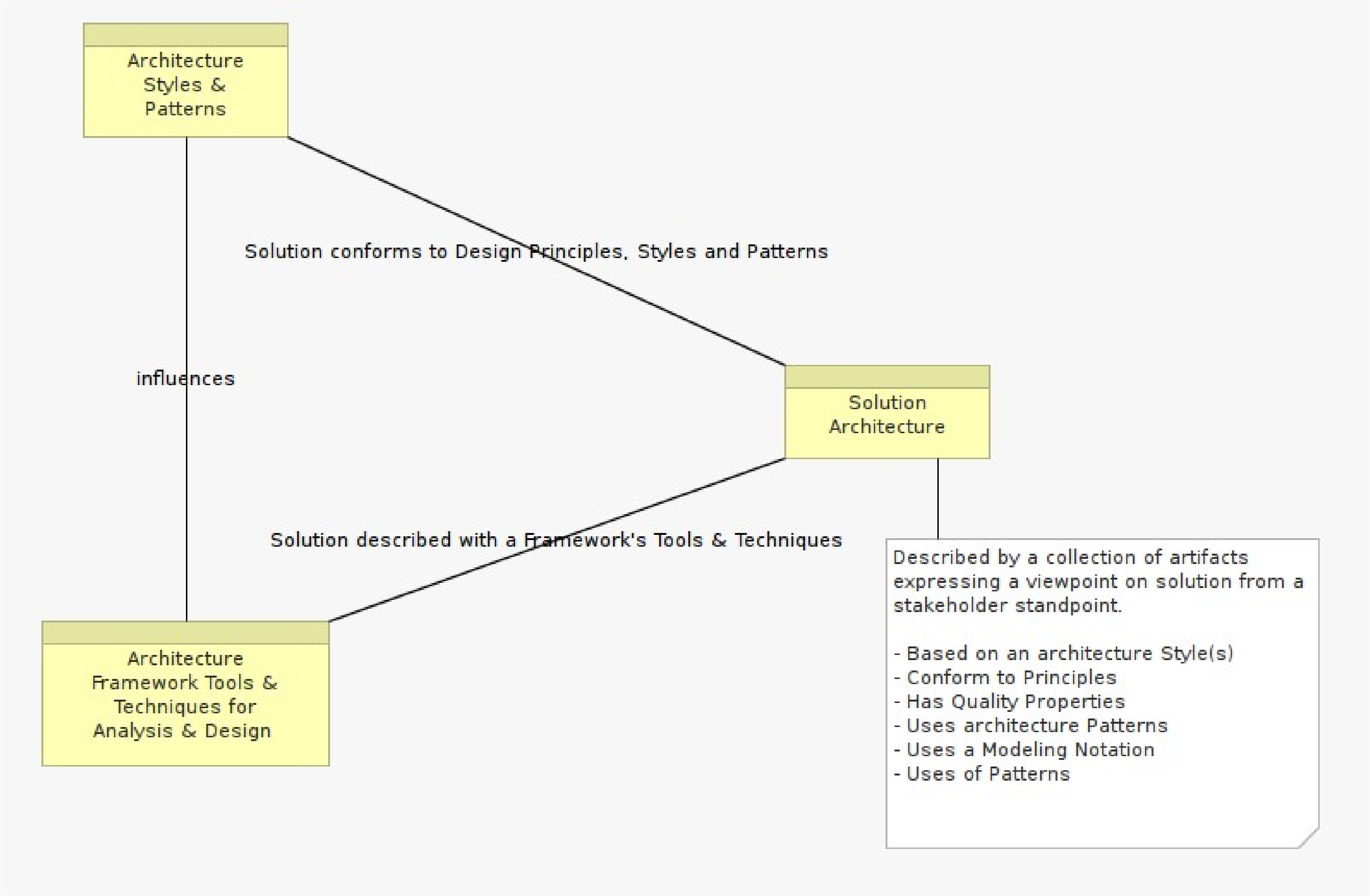
(cc) BY-NC-SA

Introduction


(2011) ISO/IEC/IEEE 42010:2011
(2011) TOGAF™ 9.1
(2011) Zachman Framework v3.0
(2011) RM-ODP ISO-EnfVP
(2010) FEAF v1.0
(2009) TOGAF™ 9.0
(2006) IEEE 1471 is an ISO Standard
(2005) MetaGroup/Gartner AF Research
(2004) Enterprise Unified Process (EUP) v5.0
(2003) E-VAP™
(2003) TOGAF™ 8.0 (Enterprise Edition)
(2001) TOGAF™ 7.0 (Technical Edition)
(2000) IEEE 1471 AF Standard
(1999) Visual Architecture Process™ (VAP) v1.0
(1999) Federal Enterprise Architecture Framework (FEAF)
(1998) Rational Unified Process (RUP) v5.0
(1998) Zachman Framework v2.0
(1996) Rational Objectory Process (ROP) v4.0
(1996) RM-ODP
(1992) Zachman Framework v1.0
(1988) Objectory v1.0
(1987) Zachman Framework for IS Architecture
2015
2011
2010
2005
2000
1995
1990
1985
State of the Art in 2016
Purpose
Use of Framework
Characteristics
Definition
Rationale
Introduction
Architecture Styles & Patterns
Solution conforms to Design Principles, Styles and Patterns
influences
Solution Architecture
Solution described with a Framework's Tools & Techniques
Architecture Framework Tools & Techniques for Analysis & Design
Described by a collection of artifacts expressing a viewpoint on solution from a stakeholder standpoint.
- Based on an architecture Style(s)
- Conform to Principles
- Has Quality Properties
- Uses architecture Patterns
- Uses a Modeling Notation
- Uses of Patterns
What is a Analytical Framework?

(cc) BY-NC-SA

What is an Analytical Framework?


Architecture Styles & Patterns
Solution conforms to Design Principles, Styles and Patterns
influences
Solution Architecture
Solution described with a Framework's Tools & Techniques
Architecture Framework Tools & Techniques for Analysis & Design
Described by a collection of artifacts expressing a viewpoint on solution from a stakeholder standpoint.
- Based on an architecture Style(s)
- Conform to Principles
- Has Quality Properties
- Uses architecture Patterns
- Uses a Modeling Notation
- Uses of Patterns

cc BY-NC-SA

Rationale

It is difficult to compare and discuss architecture options without a common frame of reference providing the basic semantics of a common language.

Any Transfer of knowledge requires dialog.

Dialog requires a common language to describe concepts.

Architecture Frameworks provide analytical Tools and Techniques to help Architects understand the "Problem to Solve" and Design answering Solutions.

Frameworks provide a common base of reference for architects, making it easy to share and transfer best practices.

(cc) BY-NC-SA

Definition

An Analytical Framework is a skeletal support used as the basis for something being constructed; a structure for supporting (or enclosing) something else.

It can loosely be defined as a set of rules and practices that constitute a way of viewing (i.e. analysing) the reality of the Business Enterprise.

Frameworks recommend standardized analysis techniques, tools, terminology, methods, standards and design protocols.

Analytical Frameworks focus on the problematic of:

(1.) Management of complexity, ...and cost implications,

(2.) Ease of change, ...and cost implications,

(3.) Alignment of Business & IT expectations, ... and cost implications.

(cc) BY-NC-SA

Characteristics

Core to an Architectural Framework mandate is to provide a PROVEN:

- a way for the Enterprise to handle change, while consolidating and capitalizing on its IT assets and steadily increasing its IT maturity.

- model for: (a.) capturing, (b.) organizing, and (c.) representing the Architecture of Business Functions, Information & Systems in the Enterprise.

- exhaustively describe all the aspects of a good Architectural Solution

- enumerate commonly known problems and pitfalls into a logical Framework.

(cc) BY-NC-SA

Use of Framework

On its own, and while extremely useful, an Analytical Framework simply embodies a set of conceptual guidelines.

Architects adjust its aspects to the needs and characteristics of the problem and solution design.

cc BY-NC-SA

Purpose

A customizable base of reference to adjust to the specifics of a Solution Architecture.

To propose a uniformed approach for the discovery, identification, sequencing, planning, implementation, integration of the "building blocks" of a solution.

(1.) How Investigate a Problem, thinking problem complexity,

(2.) How to create a Solution (i.e. using classifications, viewpoints)

(3.) How to manage change, technical debt, encouraging re-use, discouraging re-work and avoiding waste.

Consistency of skill across Architects

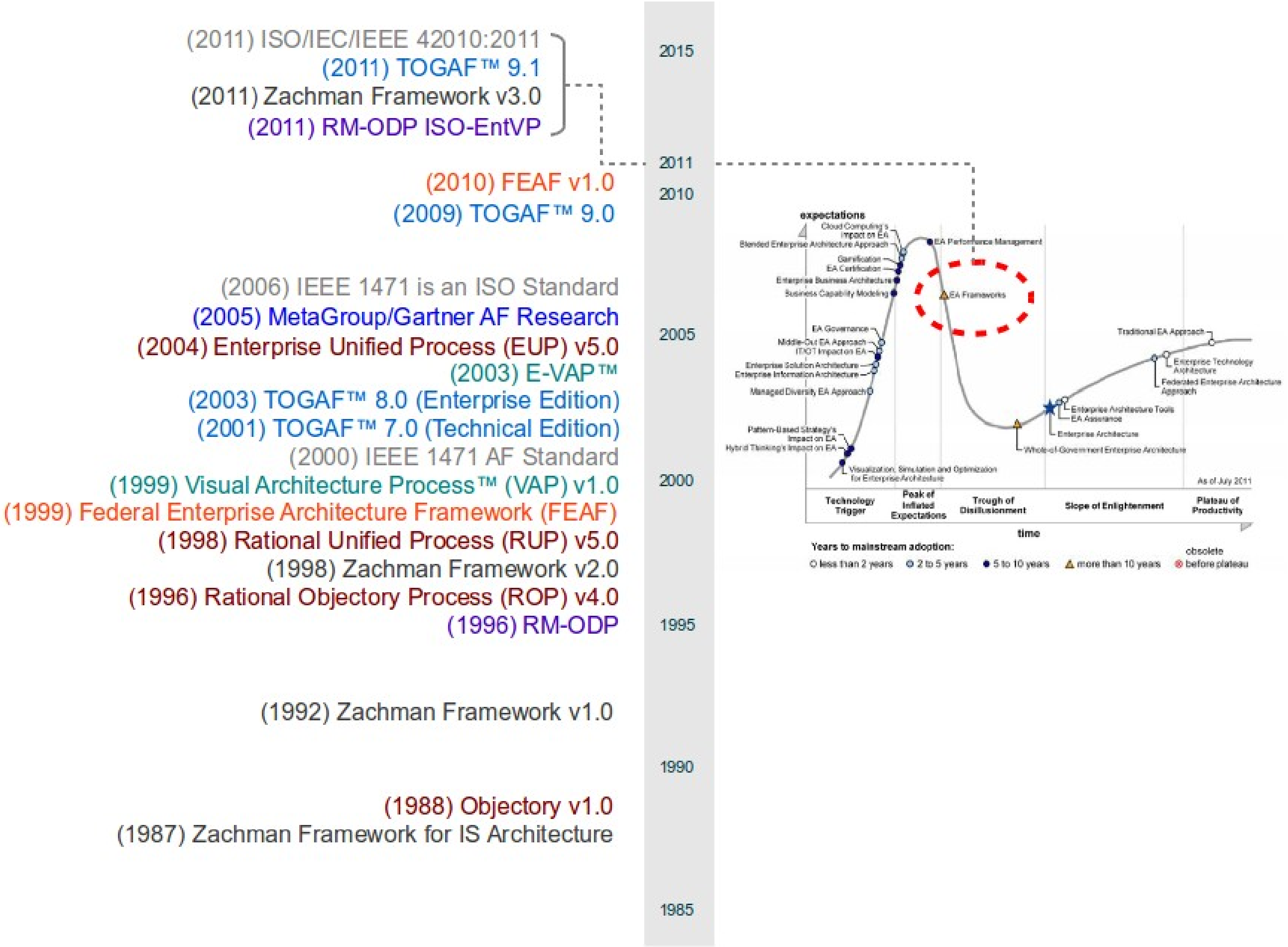
...in the due diligence and approach to business problematic and solution definition (i.e. methodology & governance triggers).

...in the use of standard deliverables issued from standardized analysis and design approaches

...avoiding the creation of "stovepipe" Solutions, that do not interact with or build on other applications, are difficult to enhance, creating "entropy".

(cc) BY-NC-SA

State of the Art in 2016



(2011) ISO/IEC/IEEE 42010:2011
(2011) TOGAF™ 9.1
(2011) Zachman Framework v3.0
(2011) RM-ODP ISO-EntVP

(2010) FEAF v1.0
(2009) TOGAF™ 9.0

(2006) IEEE 1471 is an ISO Standard
(2005) MetaGroup/Gartner AF Research
(2004) Enterprise Unified Process (EUP) v5.0
(2003) E-VAP™
(2003) TOGAF™ 8.0 (Enterprise Edition)
(2001) TOGAF™ 7.0 (Technical Edition)
(2000) IEEE 1471 AF Standard
(1999) Visual Architecture Process™ (VAP) v1.0
(1999) Federal Enterprise Architecture Framework (FEAF)
(1998) Rational Unified Process (RUP) v5.0
(1998) Zachman Framework v2.0
(1996) Rational Objectory Process (ROP) v4.0
(1996) RM-ODP

(1992) Zachman Framework v1.0

(1988) Objectory v1.0
(1987) Zachman Framework for IS Architecture

2015
2011
2010
2005
2000
1995
1990
1985
expectations
Cloud Computing's Impact on EA
Blended Enterprise Architecture Approach
Gamification
EA Certification
Enterprise Business Architecture
Business Capability Modeling
EA Performance Management
EA Frameworks
EA Governance
Middle-Out EA Approach
IT/OT Impact on EA
Enterprise Solution Architecture
Enterprise Information Architecture
Managed Diversity EA Approach
Pattern-Based Strategy's Impact on EA
Hybrid Thinking's Impact on EA
Visualization, Simulation and Optimization for Enterprise Architecture
Traditional EA Approach
Enterprise Technology Architecture
Federated Enterprise Architecture Approach
Enterprise Architecture Tools
EA Assurance
Enterprise Architecture
Whole-of-Government Enterprise Architecture
As of July 2011
Technology Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
time
Years to mainstream adoption:
○ less than 2 years
○ 2 to 5 years
● 5 to 10 years
△ more than 10 years
⊗ obsolete before plateau

(cc) BY-NC-SA

State of the Art in 2016

There are around 50+ named Architecture Frameworks in existence, tackling a variety of concerns, but few are truly distinct from one another.

Some Frameworks originate from IT and evolved to encompass the Enterprise.

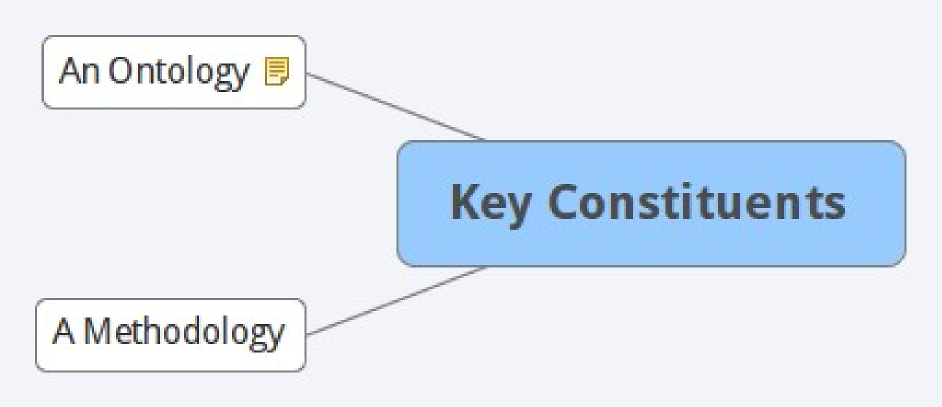
Some Frameworks are originally designed to model the Enterprise as a System.

Some Frameworks extend the modeling to Enterprise Planning.

Some Frameworks are anchored in Software Engineering & Architecture, and require adjustment to the Enterprise.

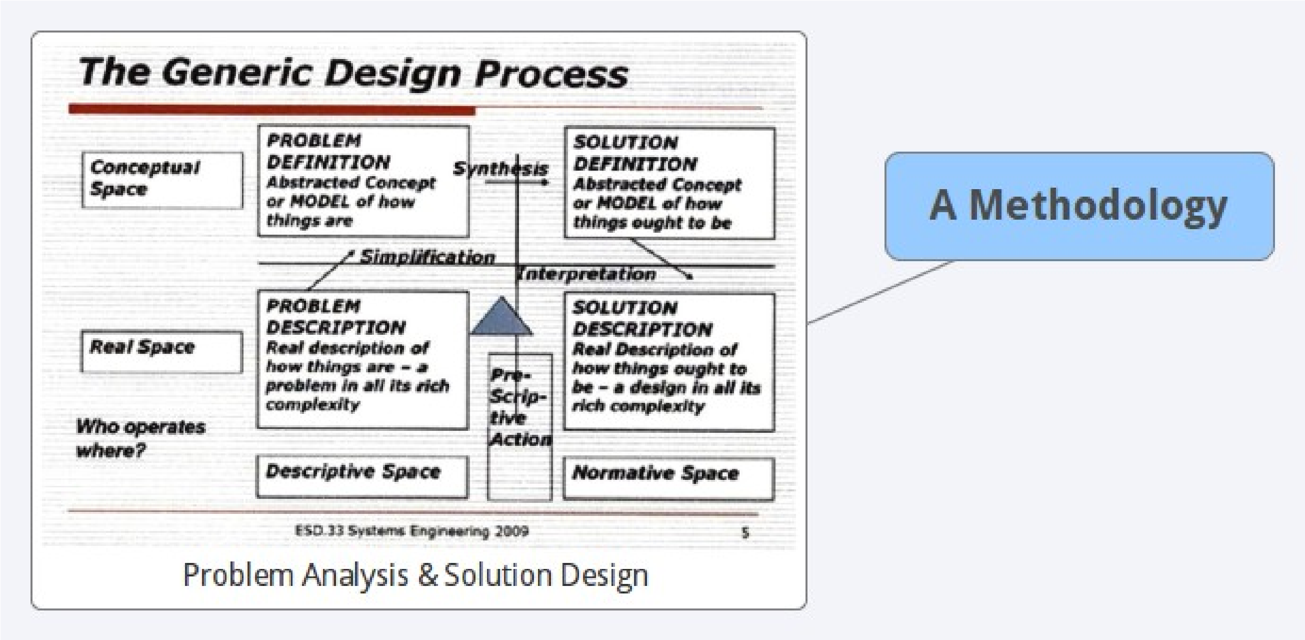
(cc) BY-NC-SA

Key Constituents



(cc) BY-NC-SA

A Methodology



The Generic Design Process

Conceptual Space

PROBLEM DEFINITION
Abstracted Concept or MODEL of how things are

Synthesis

SOLUTION DEFINITION
Abstracted Concept or MODEL of how things ought to be

Simplification

Interpretation

Real Space

PROBLEM DESCRIPTION
Real description of how things are – a problem in all its rich complexity

Pre-Scrip-tive Action

SOLUTION DESCRIPTION
Real Description of how things ought to be – a design in all its rich complexity

Who operates where?

Descriptive Space

Normative Space

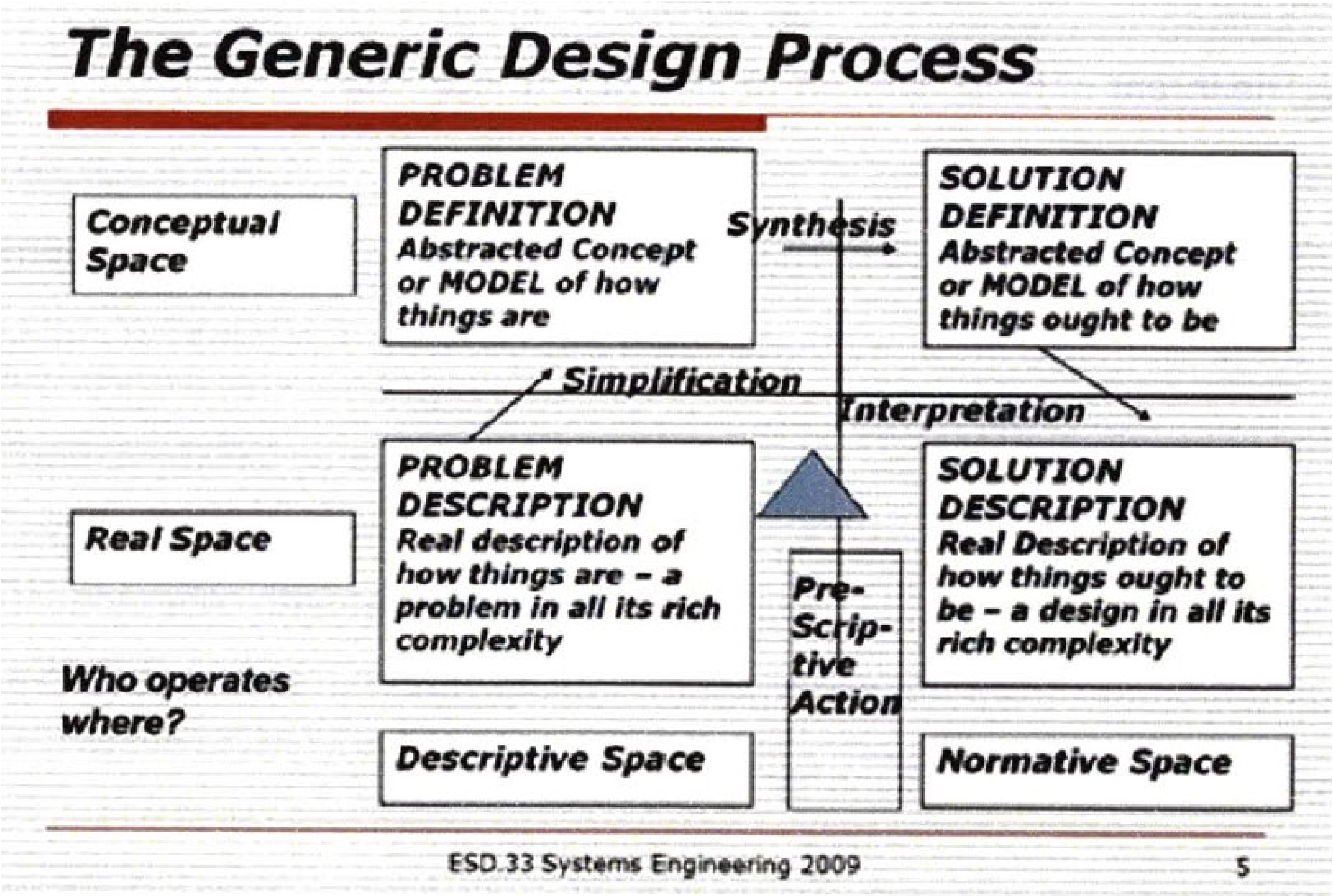
ESD.33 Systems Engineering 2009 5

Problem Analysis & Solution Design

A Methodology

(cc) BY-NC-SA

Problem Analysis & Solution Design



The Generic Design Process

Conceptual Space

PROBLEM DEFINITION
Abstracted Concept or MODEL of how things are

Synthesis

SOLUTION DEFINITION
Abstracted Concept or MODEL of how things ought to be

Simplification

Interpretation

Real Space

PROBLEM DESCRIPTION
Real description of how things are – a problem in all its rich complexity

Pre-Scrip-tive Action

SOLUTION DESCRIPTION
Real Description of how things ought to be – a design in all its rich complexity

Who operates where?

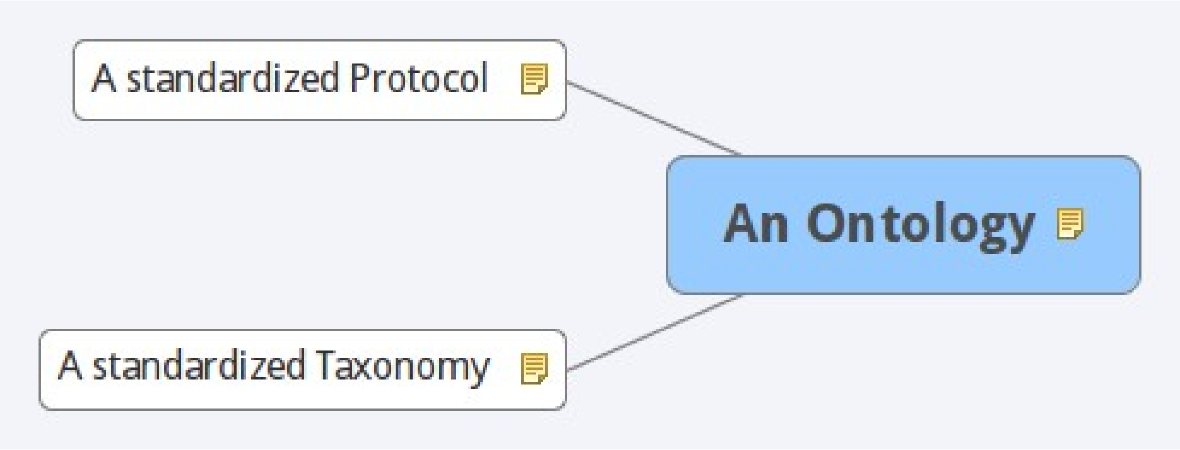
Descriptive Space

Normative Space

ESD.33 Systems Engineering 2009 5

(cc) BY-NC-SA

An Ontology



(cc) BY-NC-SA

An Ontology

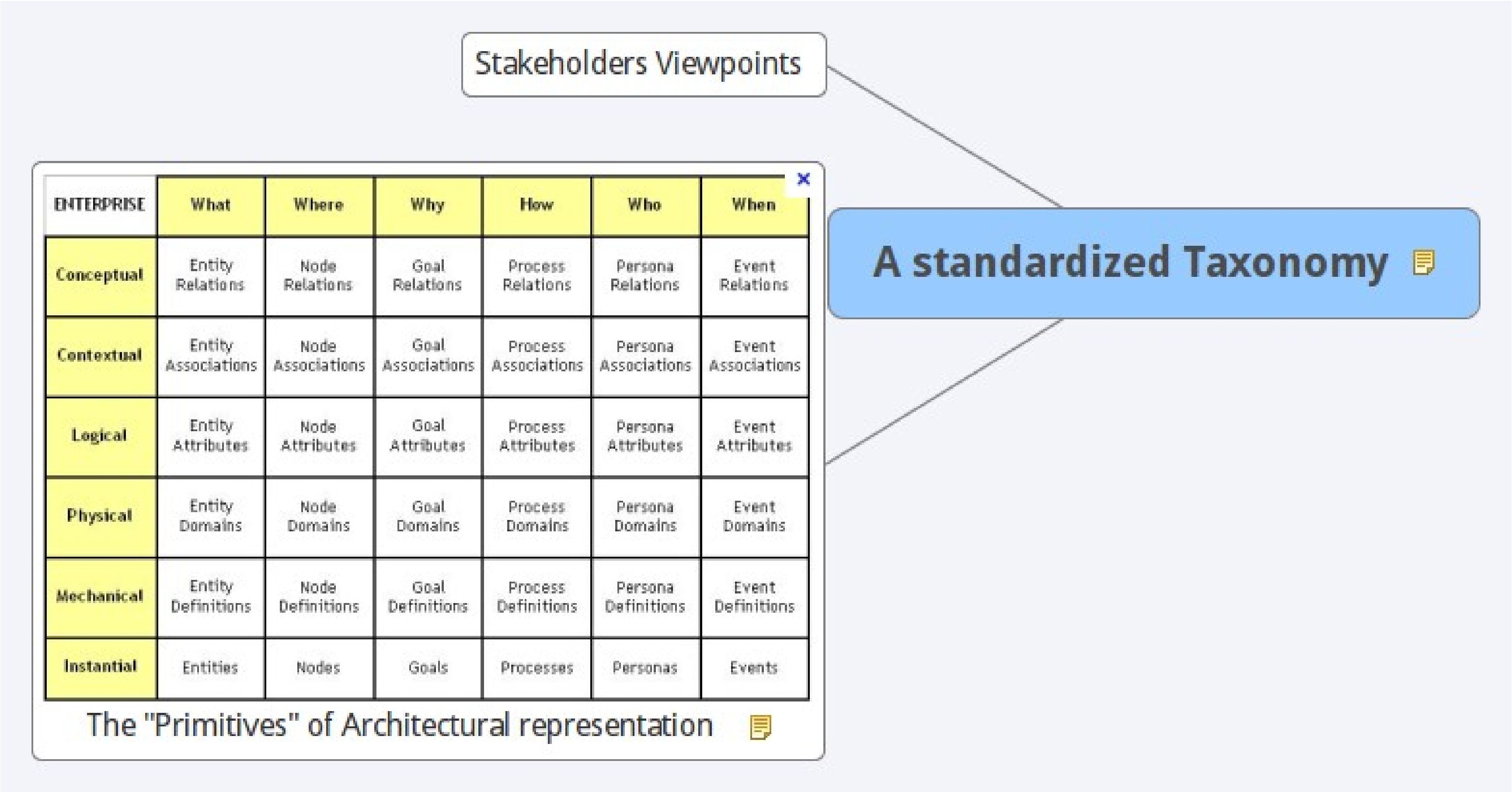
The Enterprise is seen as a multi-dimensional space and therefore need to be assessed under a set of different but interrelated angles.

It is not possible to capture the functional features and quality properties of a complex system in a single comprehensible model that is understandable by and of value to all stakeholders.

For this reason Architectural Frameworks propose a way of classifying descriptive deliverables/artifacts using different dimensions, aimed at specific stakeholders.

cc BY-NC-SA

A standardized Taxonomy



Stakeholders Viewpoints

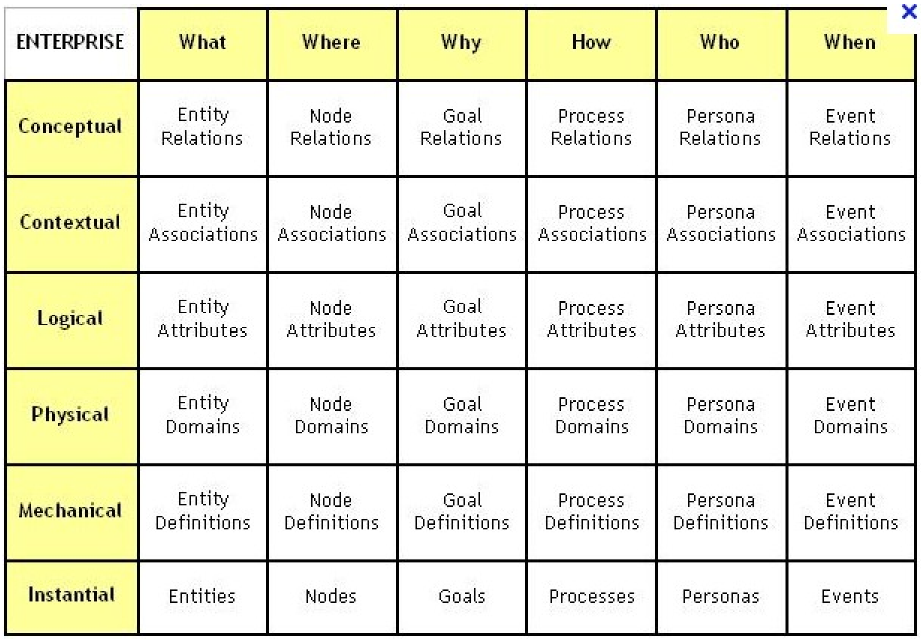
A standardized Taxonomy

ENTERPRISE	What	Where	Why	How	Who	When
Conceptual	Entity Relations	Node Relations	Goal Relations	Process Relations	Persona Relations	Event Relations
Contextual	Entity Associations	Node Associations	Goal Associations	Process Associations	Persona Associations	Event Associations
Logical	Entity Attributes	Node Attributes	Goal Attributes	Process Attributes	Persona Attributes	Event Attributes
Physical	Entity Domains	Node Domains	Goal Domains	Process Domains	Persona Domains	Event Domains
Mechanical	Entity Definitions	Node Definitions	Goal Definitions	Process Definitions	Persona Definitions	Event Definitions
Instantial	Entities	Nodes	Goals	Processes	Personas	Events

The "Primitives" of Architectural representation

(cc) BY-NC-SA

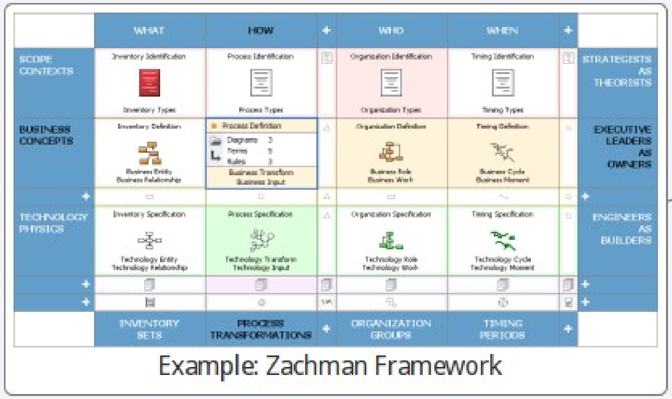
The "Primitives" of Architectural representation



ENTERPRISE	What	Where	Why	How	Who	When
Conceptual	Entity Relations	Node Relations	Goal Relations	Process Relations	Persona Relations	Event Relations
Contextual	Entity Associations	Node Associations	Goal Associations	Process Associations	Persona Associations	Event Associations
Logical	Entity Attributes	Node Attributes	Goal Attributes	Process Attributes	Persona Attributes	Event Attributes
Physical	Entity Domains	Node Domains	Goal Domains	Process Domains	Persona Domains	Event Domains
Mechanical	Entity Definitions	Node Definitions	Goal Definitions	Process Definitions	Persona Definitions	Event Definitions
Instantial	Entities	Nodes	Goals	Processes	Personas	Events

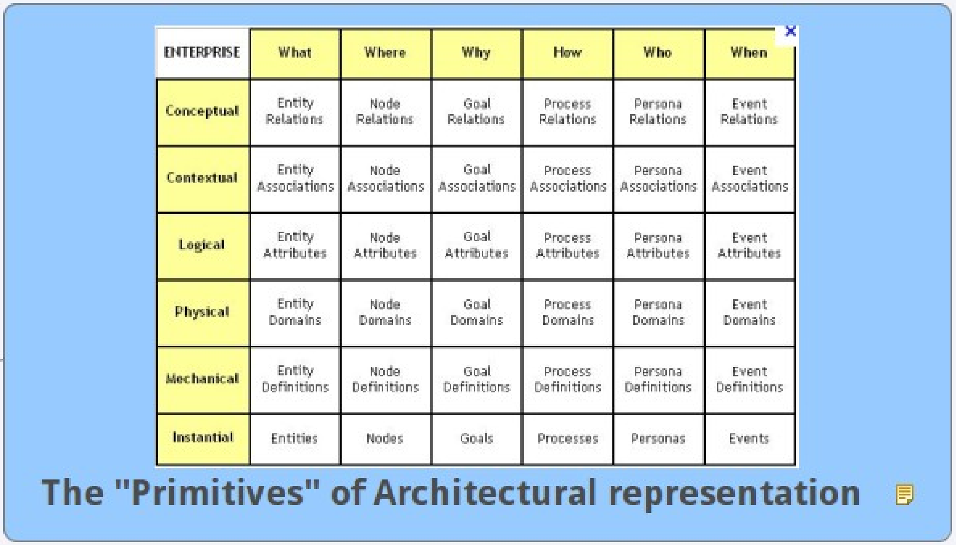
cc BY-NC-SA

The "Primitives" of Architectural representation



	WHAT	HOW	+	WHO	WHEN	+	
SCOPE CONTEXTS	Inventory Identification Inventory Types	Process Identification Process Types		Organization Identification Organization Types	Timing Identification Timing Types		STRATEGISTS AS THEORISTS
BUSINESS CONCEPTS	Inventory Definition Business Entity Business Relationship	Process Definition Diagrams 3 Terms 5 Rules 3 Business Transform Business Input		Organization Definition Business Role Business Work	Timing Definition Business Cycle Business Moment		EXECUTIVE LEADERS AS OWNERS
+							+
TECHNOLOGY PHYSICS	Inventory Specification Technology Entity Technology Relationship	Process Specification Technology Transform Technology Input		Organization Specification Technology Role Technology Work	Timing Specification Technology Cycle Technology Moment		ENGINEERS AS BUILDERS
+							+
+							+
	INVENTORY SETS	PROCESS TRANSFORMATIONS	+	ORGANIZATION GROUPS	TIMING PERIODS	+	

Example: Zachman Framework



ENTERPRISE	What	Where	Why	How	Who	When
Conceptual	Entity Relations	Node Relations	Goal Relations	Process Relations	Persona Relations	Event Relations
Contextual	Entity Associations	Node Associations	Goal Associations	Process Associations	Persona Associations	Event Associations
Logical	Entity Attributes	Node Attributes	Goal Attributes	Process Attributes	Persona Attributes	Event Attributes
Physical	Entity Domains	Node Domains	Goal Domains	Process Domains	Persona Domains	Event Domains
Mechanical	Entity Definitions	Node Definitions	Goal Definitions	Process Definitions	Persona Definitions	Event Definitions
Instantial	Entities	Nodes	Goals	Processes	Personas	Events

The "Primitives" of Architectural representation

cc BY-NC-SA

The "Primitives" of Architectural representation

An Enterprise is a complex System and as such is composed of many Domain Specialists.

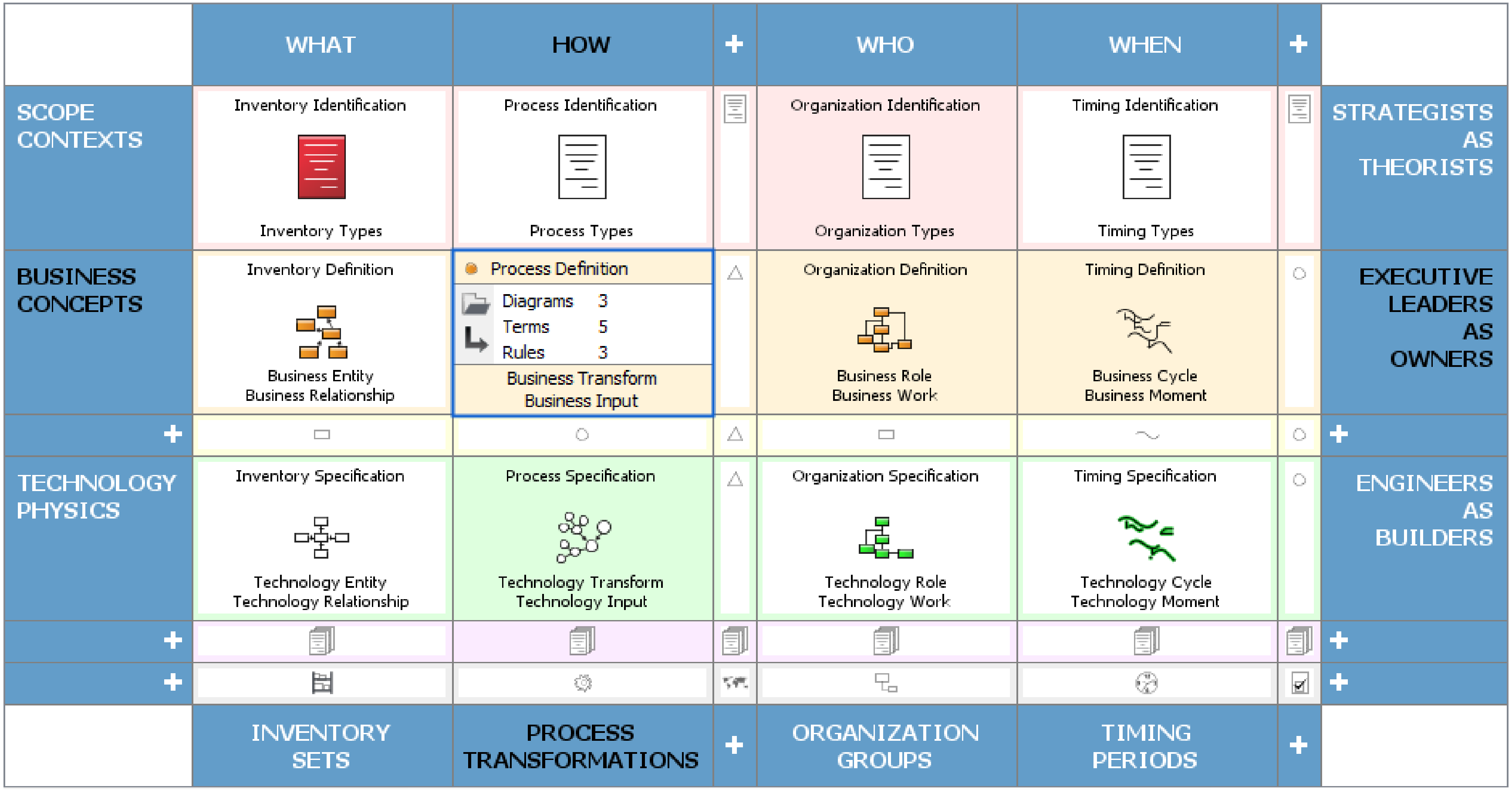
Specialists have a perspective on things, a natural appreciation of a given subject of their field).

Viewpoints help to characterize and distinguish Architectural representations, and make them understandable to Specialists.

The intersection of 2 lines qualifies the simplest expression of an architectural view that addresses a precise concern.

(cc) BY-NC-SA

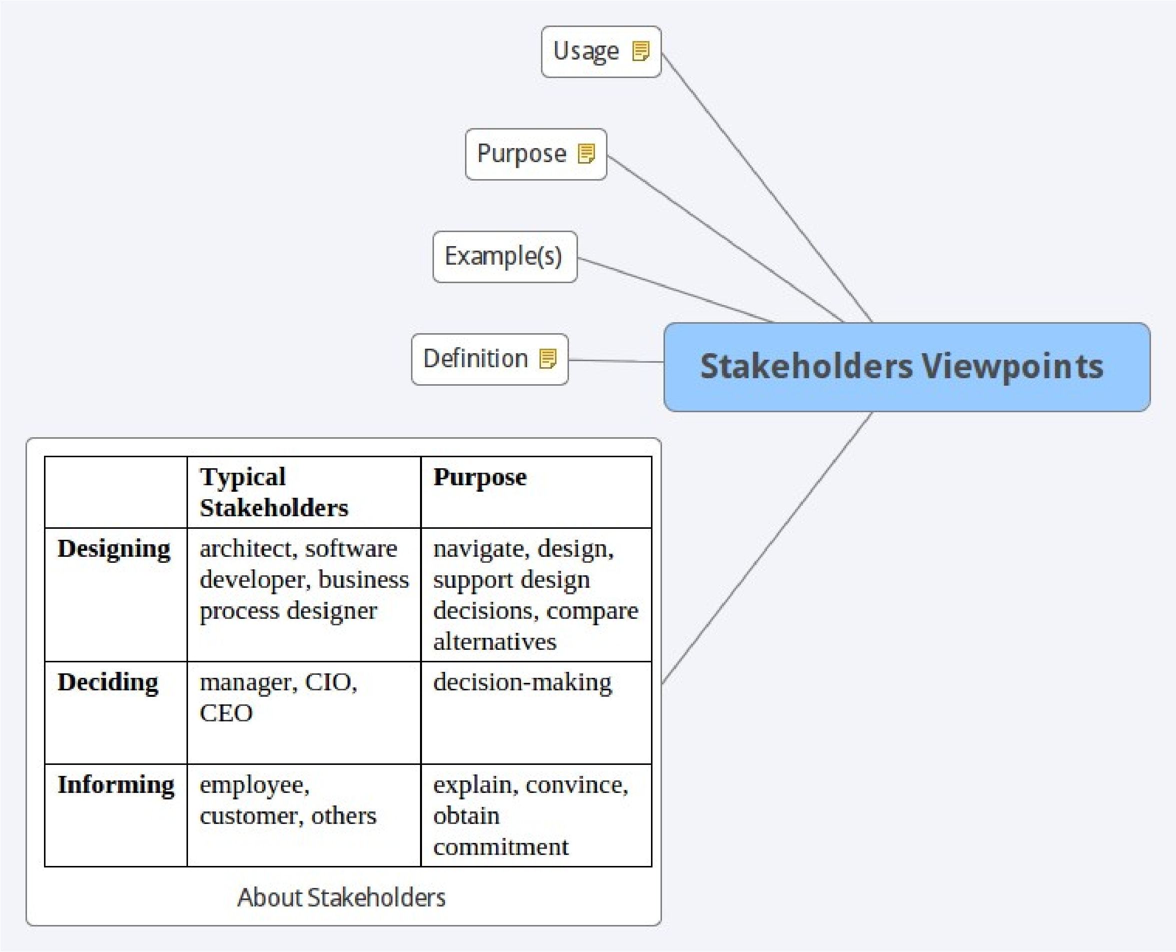
Example: Zachman Framework



	WHAT	HOW	+	WHO	WHEN	+	
SCOPE CONTEXTS	Inventory Identification Inventory Types	Process Identification Process Types		Organization Identification Organization Types	Timing Identification Timing Types		STRATEGISTS AS THEORISTS
BUSINESS CONCEPTS	Inventory Definition Business Entity Business Relationship	Process Definition Diagrams 3 Terms 5 Rules 3 Business Transform Business Input	△	Organization Definition Business Role Business Work	Timing Definition Business Cycle Business Moment	○	EXECUTIVE LEADERS AS OWNERS
+	▭	○	△	▭	∼	○	+
TECHNOLOGY PHYSICS	Inventory Specification Technology Entity Technology Relationship	Process Specification Technology Transform Technology Input	△	Organization Specification Technology Role Technology Work	Timing Specification Technology Cycle Technology Moment	○	ENGINEERS AS BUILDERS
+							+
+						☑	+
	INVENTORY SETS	PROCESS TRANSFORMATIONS	+	ORGANIZATION GROUPS	TIMING PERIODS	+	

cc BY-NC-SA

Stakeholders Viewpoints

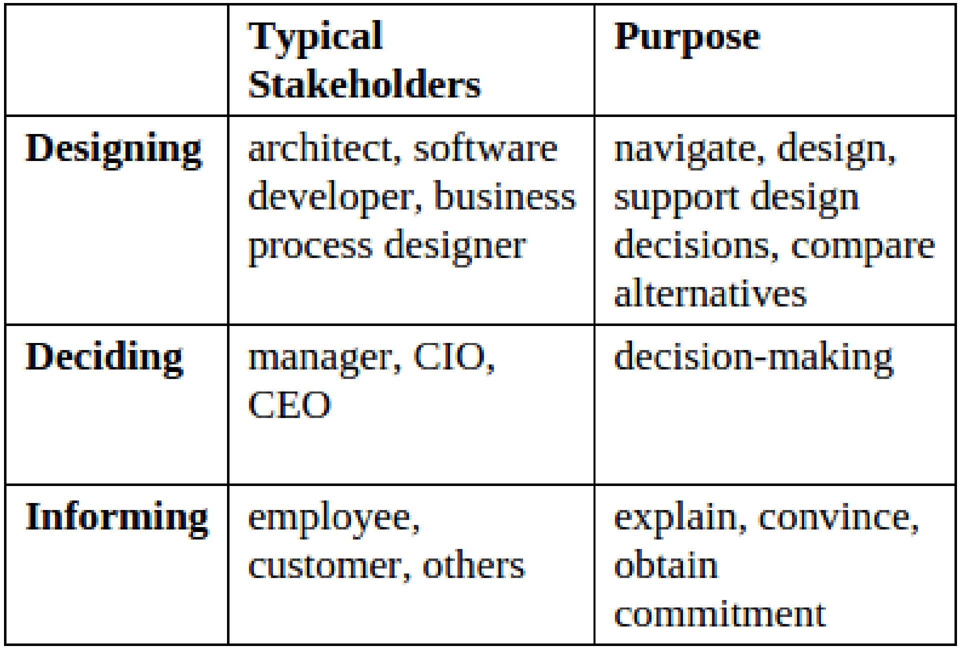


	Typical Stakeholders	Purpose
Designing	architect, software developer, business process designer	navigate, design, support design decisions, compare alternatives
Deciding	manager, CIO, CEO	decision-making
Informing	employee, customer, others	explain, convince, obtain commitment

About Stakeholders

(cc) BY-NC-SA

About Stakeholders



	Typical Stakeholders	Purpose
Designing	architect, software developer, business process designer	navigate, design, support design decisions, compare alternatives
Deciding	manager, CIO, CEO	decision-making
Informing	employee, customer, others	explain, convince, obtain commitment

cc BY-NC-SA

Definition

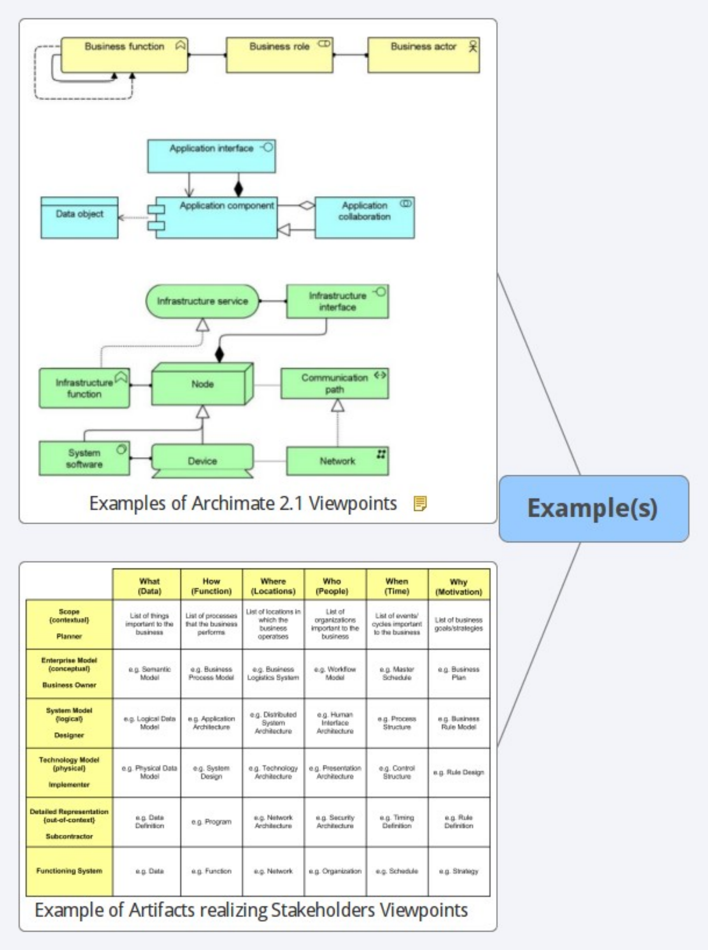
Each Viewpoint determine the language (including notations, model, or product types) to be used to represent the architecture from the standpoint of a stakeholder.

Viewpoints express how to model a "view" of the solution architecture that speaks to stakeholder.

Viewpoints are the building blocks of a library of Templates that can be used off the shelf to guide the creation of an Architecture Solution (ex. TOGAF 9.1/Archimate 2.1 Viewpoints).

(cc) BY-NC-SA

Example(s)



Examples of Archimate 2.1 Viewpoints

	What (Data)	How (Function)	Where (Locations)	Who (People)	When (Time)	Why (Motivation)
Scope (contextual) Planner	List of things important to the business	List of processes that the business performs	List of locations in which the business operatses	List of organizations important to the business	List of events/ cycles important to the business	List of business goals/strategies
Enterprise Model (conceptual) Business Owner	e.g. Semantic Model	e.g. Business Process Model	e.g. Business Logistics System	e.g. Workflow Model	e.g. Master Schedule	e.g. Business Plan
System Model (logical) Designer	e.g. Logical Data Model	e.g. Application Architecture	e.g. Distributed System Architecture	e.g. Human Interface Architecture	e.g. Process Structure	e.g. Business Rule Model
Technology Model (physical) Implementer	e.g. Physical Data Model	e.g. System Design	e.g. Technology Architecture	e.g. Presentation Architecture	e.g. Control Structure	e.g. Rule Design
Detailed Representation (out-of-context) Subcontractor	e.g. Data Definition	e.g. Program	e.g. Network Architecture	e.g. Security Architecture	e.g. Timing Definition	e.g. Rule Definition
Functioning System	e.g. Data	e.g. Function	e.g. Network	e.g. Organization	e.g. Schedule	e.g. Strategy

Example of Artifacts realizing Stakeholders Viewpoints

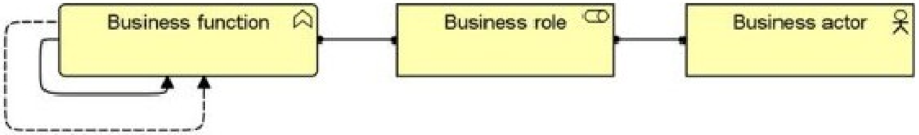
cc BY-NC-SA

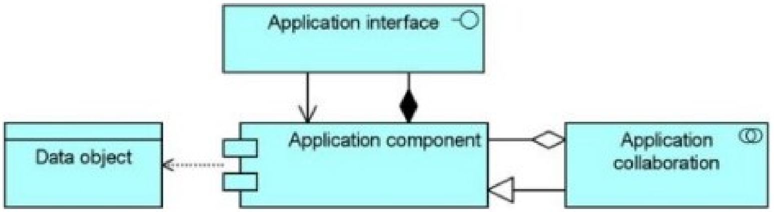
Example of Artifacts realizing Stakeholders Viewpoints

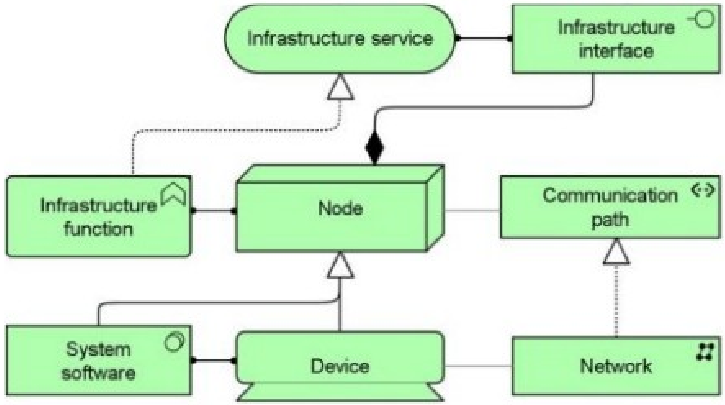
	What (Data)	**How (Function)**	**Where (Locations)**	**Who (People)**	**When (Time)**	**Why (Motivation)**
Scope {contextual} Planner	List of things important to the business	List of processes that the business performs	List of locations in which the business operatses	List of organizations important to the business	List of events/ cycles important to the business	List of business goals/strategies
Enterprise Model {conceptual} Business Owner	e.g. Semantic Model	e.g. Business Process Model	e.g. Business Logistics System	e.g. Workflow Model	e.g. Master Schedule	e.g. Business Plan
System Model {logical} Designer	e.g. Logical Data Model	e.g. Application Architecture	e.g. Distributed System Architecture	e.g. Human Interface Architecture	e.g. Process Structure	e.g. Business Rule Model
Technology Model {physical} Implementer	e.g. Physical Data Model	e.g. System Design	e.g. Technology Architecture	e.g. Presentation Architecture	e.g. Control Structure	e.g. Rule Design
Detailed Representation {out-of-context} Subcontractor	e.g. Data Definition	e.g. Program	e.g. Network Architecture	e.g. Security Architecture	e.g. Timing Definition	e.g. Rule Definition
Functioning System	e.g. Data	e.g. Function	e.g. Network	e.g. Organization	e.g. Schedule	e.g. Strategy

(cc) BY-NC-SA

Examples of Archimate 2.1 Viewpoints







(cc) BY-NC-SA

Examples of Archimate 2.1 Viewpoints

Business Function Viewpoint: shows the main business functions of an organization and their relationships in terms of the flows of information.

Application Structure Viewpoint: shows the structure of one or more applications or components.

Infrastructure Usage Viewpoint: shows how one or more applications are realized on the infrastructure.

...etc.

(cc) BY-NC-SA

Purpose

To give architect a key (or legend) to understanding the notations in views of that type.

Viewpoints are an important way of bringing much-needed structure and consistency to what was in the past a fairly unstructured activity.

Viewpoints are defined in a uniform manner can be documented, put on the shelf, reused, and improved upon across the community of architects.

cc BY-NC-SA

Usage

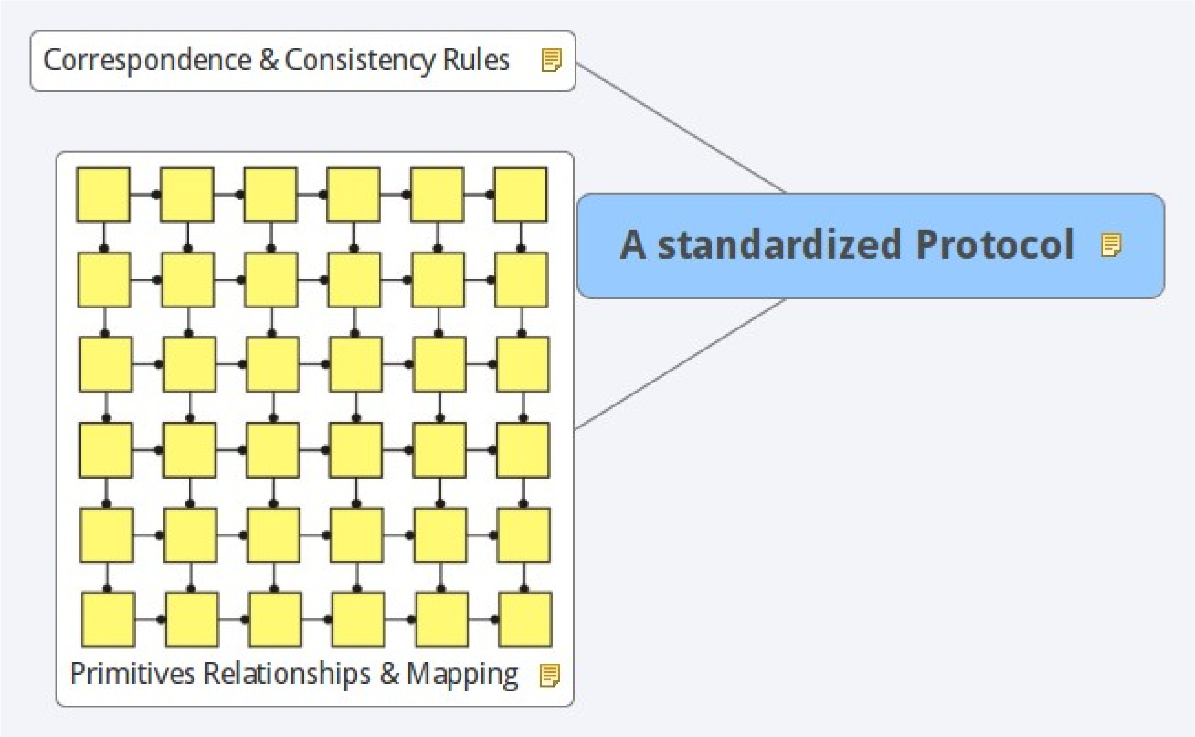
The identification and selection of stakeholders, concerns, and viewpoints, and the construction of model artifacts, is the responsibility of the architect, in association with the stakeholders.

Frameworks do not dictate to Architects which Viewpoints to use, because there is no consensus on which aspects of the problem or solution are important, and because systems vary widely in purpose.

(cc) BY-NC-SA

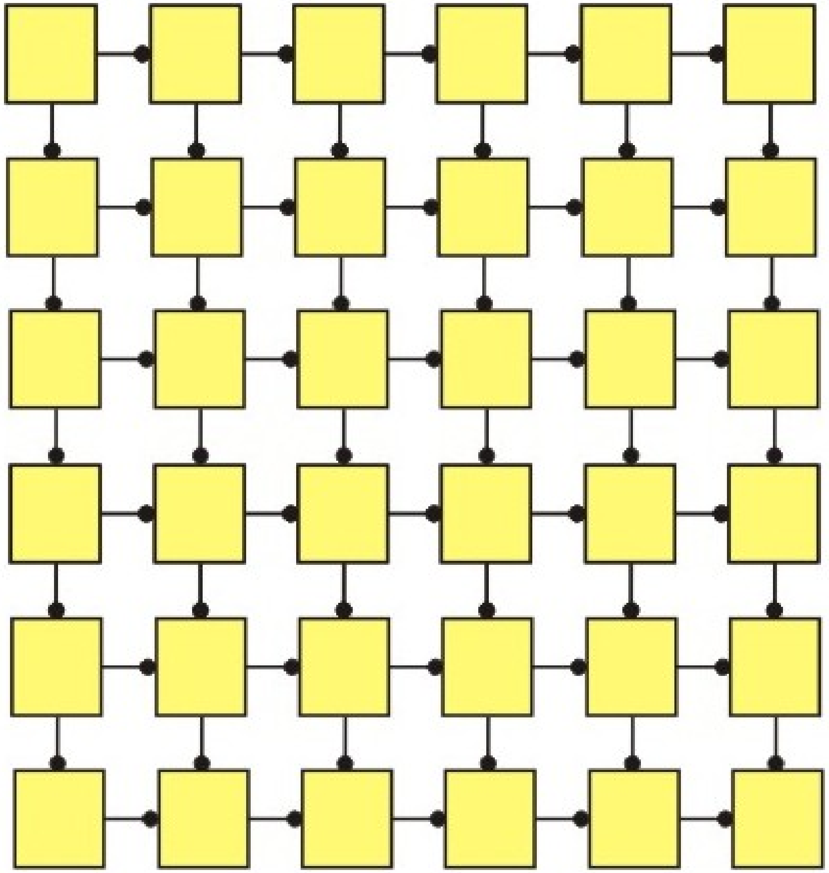
A standardized Protocol

Protocols address problems that typically arise whenever multiple Views are used to generate an Architectural Description.


Correspondence & Consistency Rules
A standardized Protocol
Primitives Relationships & Mapping

(cc) BY-NC-SA

Primitives Relationships & Mapping



Taxonomies implicitly logical relationships between Entities.

Relationships between entities classified are one to many left to right and one to many top to bottom.

Relationships define "protocols" speaking to the order in which Entities should be (can be) created.

Arbitrary and more explicit relationships can be defined to superset the implicit two-dimensional structure.

(cc) BY-NC-SA

Correspondence & Consistency Rules

Correspondence Rules express relationships between elements within an architecture description to enforce architecture relations such as:

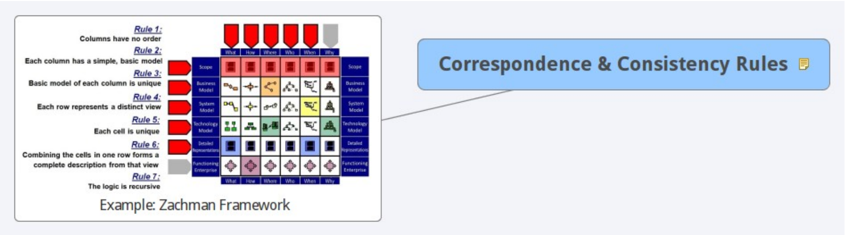
(1.) Composition,

(2.) Dependency,

(3.) Constraint and obligation,

(4.) Traceability.

Consistency Rules enforce a common level of refinement and detail of specification across Entities.



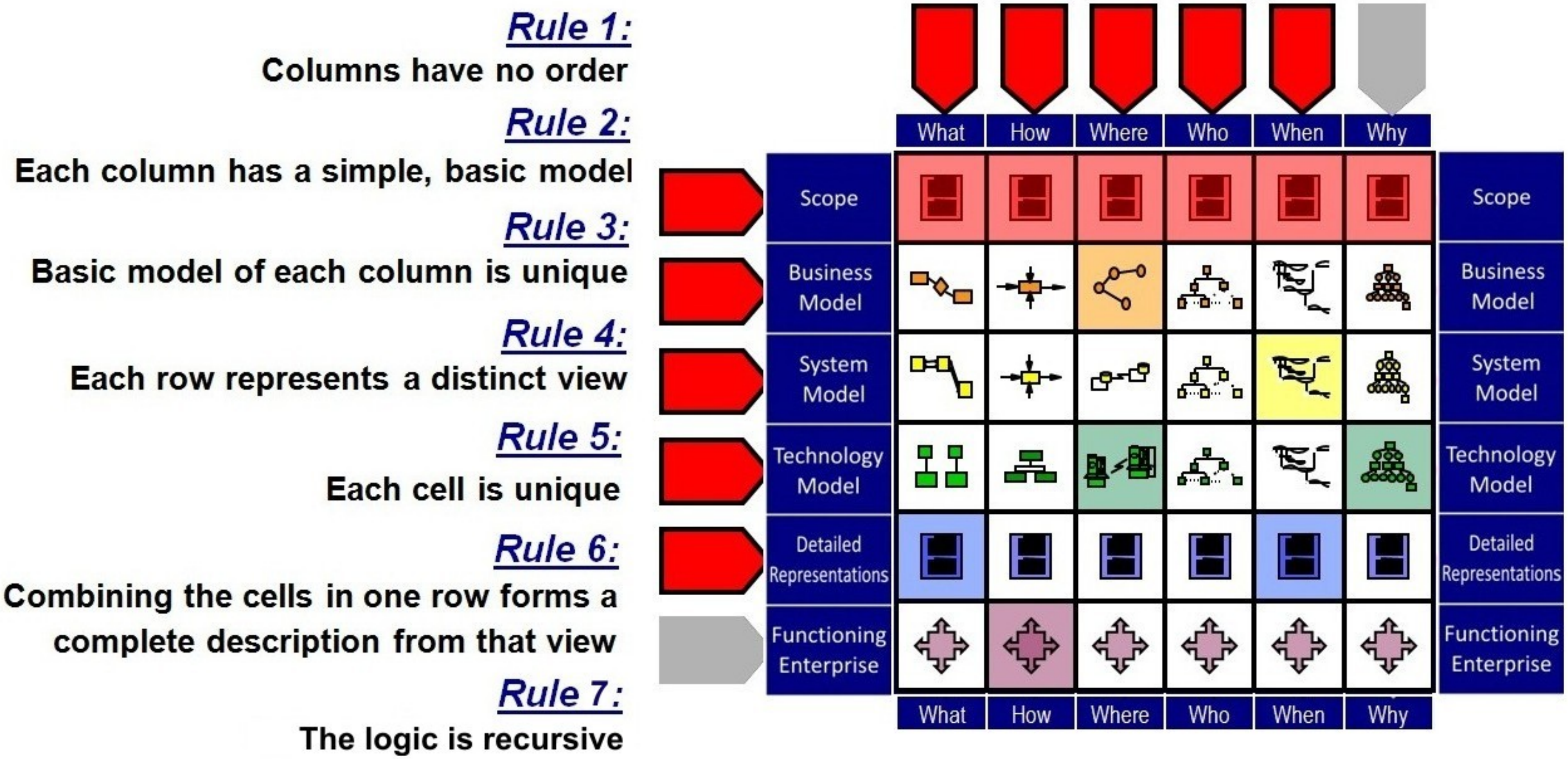
Rule 1:
Columns have no order
Rule 2:
Each column has a simple, basic model
Rule 3:
Basic model of each column is unique
Rule 4:
Each row represents a distinct view
Rule 5:
Each cell is unique
Rule 6:
Combining the cells in one row forms a complete description from that view
Rule 7:
The logic is recursive

Example: Zachman Framework

Correspondence & Consistency Rules

(cc) BY-NC-SA

Example: Zachman Framework



Rule 1:
Columns have no order

Rule 2:
Each column has a simple, basic model

Rule 3:
Basic model of each column is unique

Rule 4:
Each row represents a distinct view

Rule 5:
Each cell is unique

Rule 6:
Combining the cells in one row forms a complete description from that view

Rule 7:
The logic is recursive

(cc) BY-NC-SA

Two of the most common Frameworks in Review

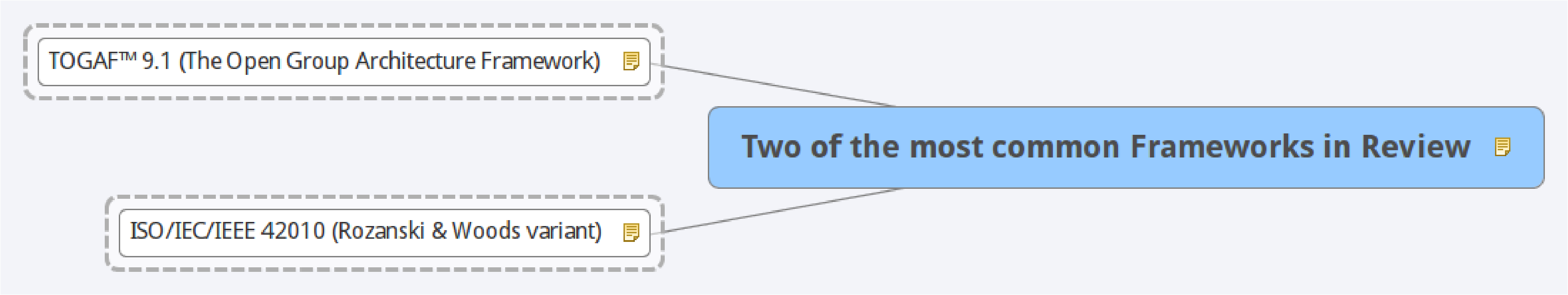
At a high-level, all Frameworks are composed of "building-blocks": (1.) a "Skeleton Structure", (2.) a "Nervous System", (3.) a "Tool-box".

They specify a widely tested and recognized set of "building blocks" to manage the Enterprise Architecture.

They propose a logical representation of the Enterprise.

They provide a set of practices to construct an Architecture.

Some Frameworks are more prescriptive than others in the Contents, Methodologies and Tools they advise.

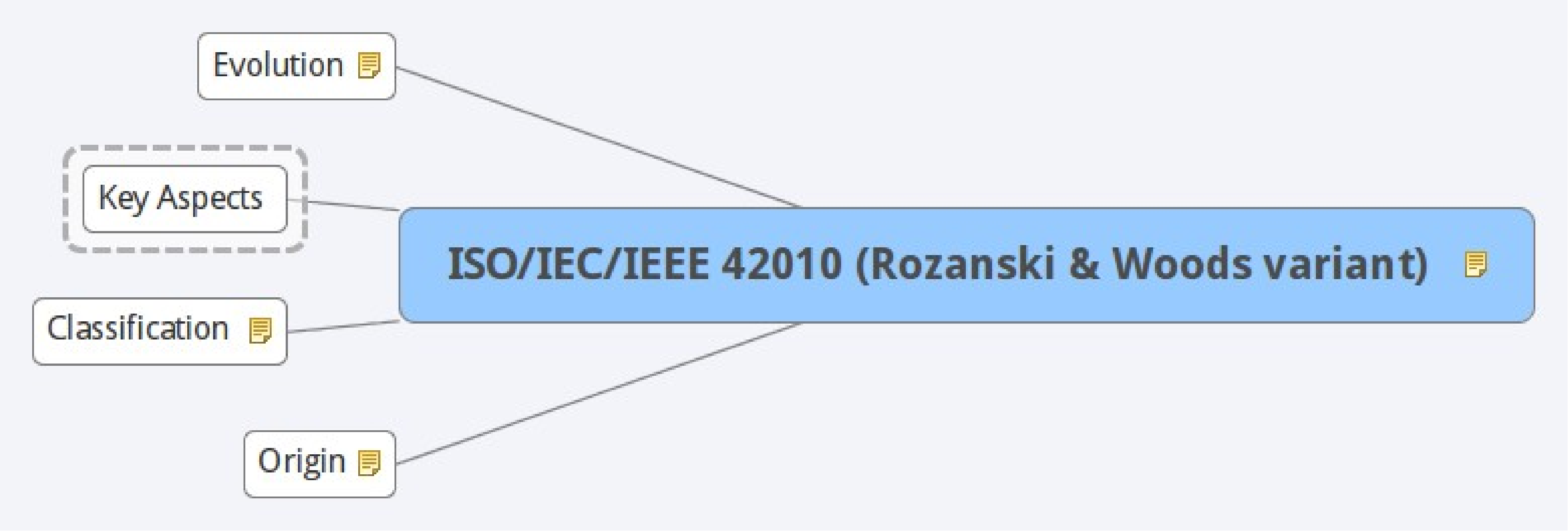

TOGAF™ 9.1 (The Open Group Architecture Framework)
Two of the most common Frameworks in Review
ISO/IEC/IEEE 42010 (Rozanski & Woods variant)

(cc) BY-NC-SA

ISO/IEC/IEEE 42010 (Rozanski & Woods variant)

A taxonomy-centric Framework, based on IEEE 1471.

The Rozanski & Woods Framework shows how a modern, IEEE/ISO compliant EA framework can be implemented from the ISO/IEC/IEEE 42010 standard.



(cc) BY-NC-SA

Origin

Originally defined in 2000 (as IEEE-1471) to provide a direction for incorporating Architectural "thinking" into IEEE standards.

IEEE 1471 was developed by the IEEE Architecture Working Group under the sponsorship of the IEEE Software Engineering Standards Committee.

The current Active Standard, published in 2011, is the result of a joint ISO and IEEE revision of the earlier IEEE Std 1471:2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.

(cc) BY-NC-SA

Classification

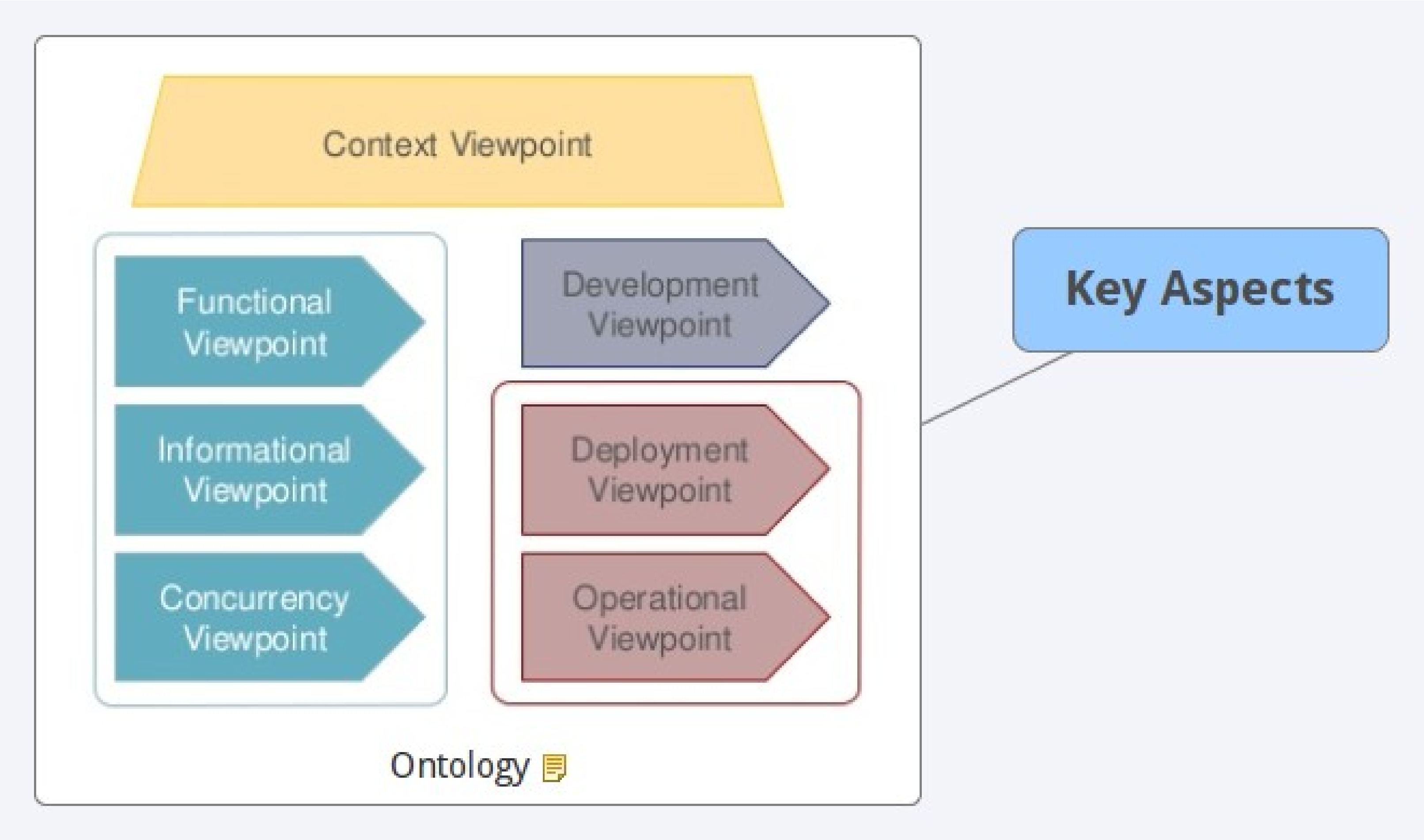
IEEE-1471 is a taxonomy-centric architecture framework.

It proposes a set of Architecture Viewpoints to codifying conventions and common practices of Architecture Description.

It provides a framework and vocabulary for talking about the constituents of an Application Architecture.

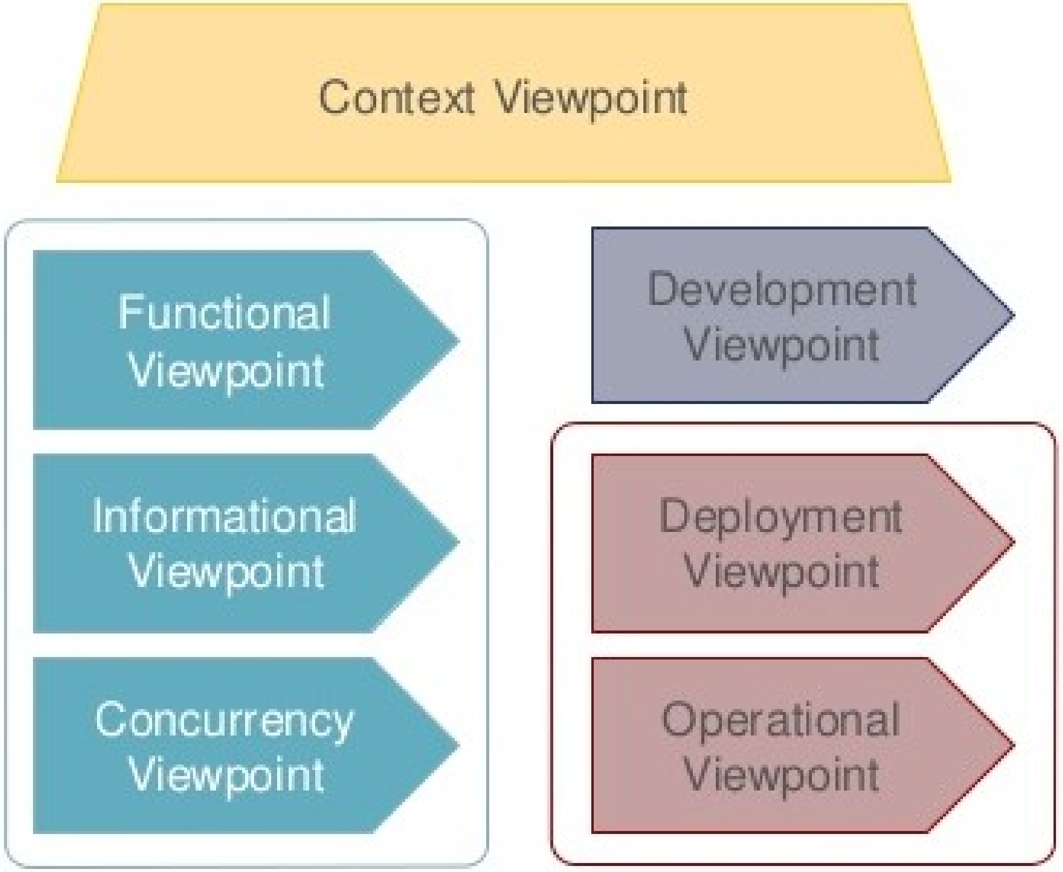
(cc) BY-NC-SA

Key Aspects



cc BY-NC-SA

Ontology



(cc) BY-NC-SA

Ontology

Context Viewpoint: Business Management, Business Analysts

Functional, Informational, Concurrency Viewpoints: System Analysts, Database designers and System Designers.

Development Viewpoint: System and Software Engineers.

Deployment and Operational Viewpoints: Administrators, Release Engineers, Operations

(cc) BY-NC-SA

Evolution

2000: The IEEE Standards Board approved IEEE 1471 for use.

2006: IEEE 1471 was adopted as an ISO standard.

2007: Publication of the Standard as ISO/IEC 42010:2007 - text identical to IEEE 1471:2000.

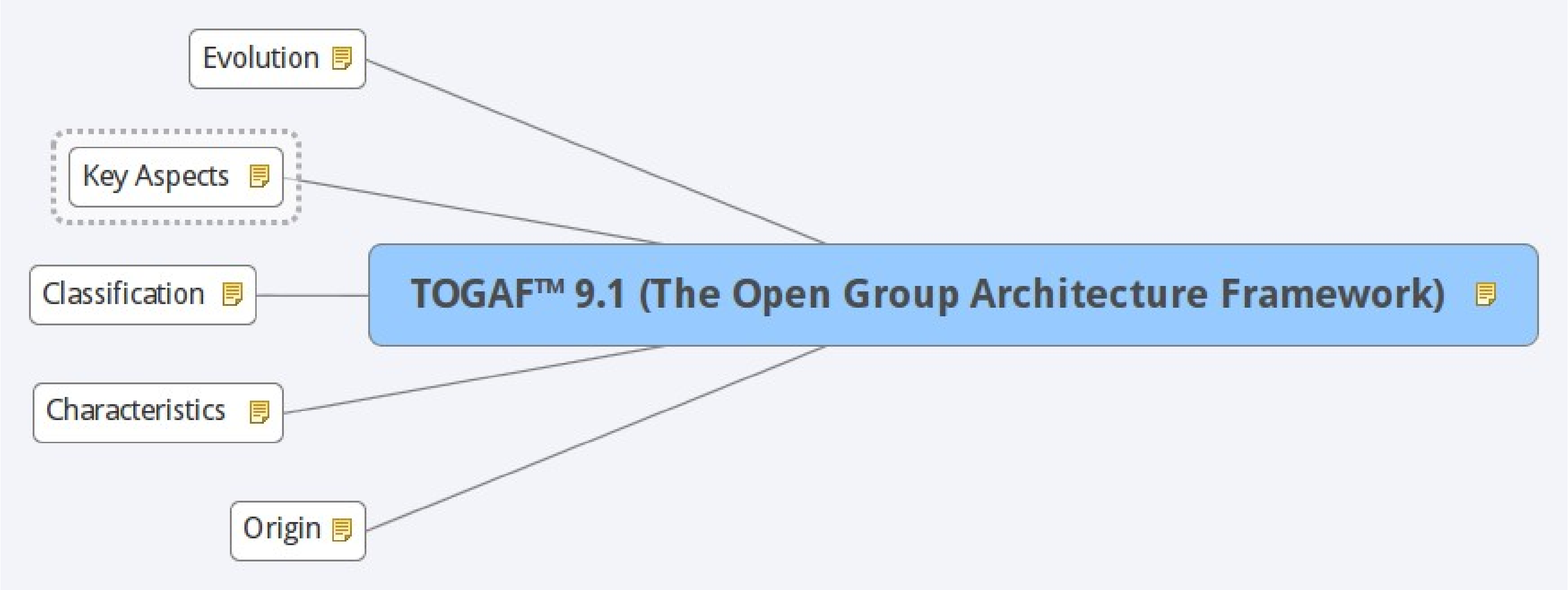
2011: ISO/IEC/IEEE 42010:2011 replaces both ISO/IEC 42010:2007 and IEEE Std 1471:2000.

(cc) BY-NC-SA

TOGAF™ 9.1 (The Open Group Architecture Framework)

A methodology-centric EA Framework.

The TOGAF Framework is an EA framework which provides a comprehensive approach to the Planning, Design, Implementation, and Governance of an Enterprise Information Architecture.



(cc) BY-NC-SA

Origin

The Open Group Architectural Framework (TOGAF) was first developed in 1995 and is based on the Department of Defense's Technical Architecture Framework for Information Management.

The TOGAF framework definition focuses on mission critical business initiatives that use IT Systems as building blocks.

It has been developed by the Architecture Forum user group within The Open Group (CapGemini, HP, HSBC, IBM, NEC, SAP, Sun Microsystems).

It is vendor neutral, Technology neutral, Tool neutral. It is currently at revision 9.1.

(cc) BY-NC-SA

Characteristics

A core and key element of TOGAF is the Architecture Development Method (ADM), which specifies a Process for creating Architectures.

TOGAF relies heavily on modularization, standardization and already existing, proven technologies and products.

TOGAF explains rules for developing good principles, rather than providing a set of architecture principles. Three levels of Principles support decision making across the entire enterprise, provide guidance of IT resources and support architecture principles for development and implementation.

(cc) BY-NC-SA

Classification

TOGAF is a Methodology-centric Architecture Framework.

(cc) BY-NC-SA

Key Aspects

The TOGAF "Content Framework" provides a loose taxonomy to categorize Architectural work-products.

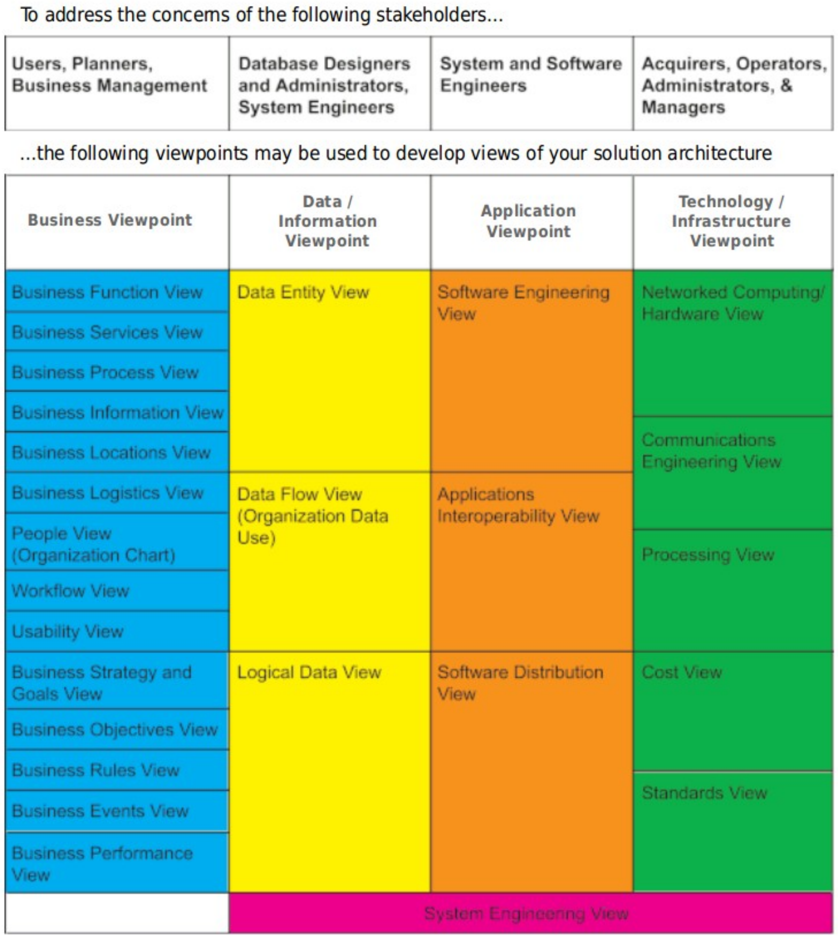
This taxonomy aims to drive consistency in the outputs from TOGAF.

It promotes reuse through consistency – a reusable “part” can be used in future architectures.

It promotes integration between architectures (and resulting system) through consistency.

(cc) BY-NC-SA

Ontology



To address the concerns of the following stakeholders…

Users, Planners, Business Management	Database Designers and Administrators, System Engineers	System and Software Engineers	Acquirers, Operators, Administrators, & Managers

…the following viewpoints may be used to develop views of your solution architecture

Business Viewpoint	Data / Information Viewpoint	Application Viewpoint	Technology / Infrastructure Viewpoint
Business Function View	Data Entity View	Software Engineering View	Networked Computing/ Hardware View
Business Services View			
Business Process View			
Business Information View			Communications Engineering View
Business Locations View			
Business Logistics View	Data Flow View (Organization Data Use)	Applications Interoperability View	
People View (Organization Chart)			Processing View
Workflow View			
Usability View			
Business Strategy and Goals View	Logical Data View	Software Distribution View	Cost View
Business Objectives View			
Business Rules View			Standards View
Business Events View			
Business Performance View			
	System Engineering View		

(cc) BY-NC-SA

Methodology

The TOGAF Methodology is composed of: (1.) The "Continuum of Architectures", (2.) The Architecture Development Process (ADM).

The Architecture Development Process (ADM) aims to be customizable for a given Organization and Project.

It allows requirements to change, and accommodates change.

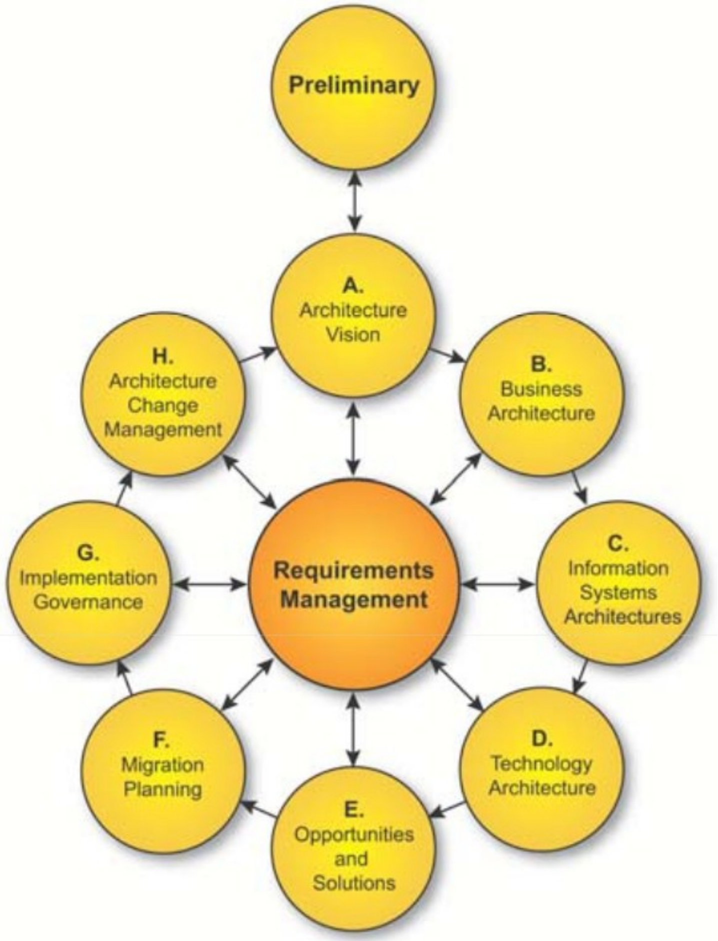
It is iterative over the whole process, between phases, and within phases.

By doing so, it avoids conflict with other processes by providing flexibility between its phases.

At any point of an IT initiative, the ADM can detour into from Requirements Change, and back out to the relevant phase.

BY-NC-SA

ADM Phases


Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

(cc) BY-NC-SA

Evolution

2001: TOGAF 7 ("Technical Edition") is published.

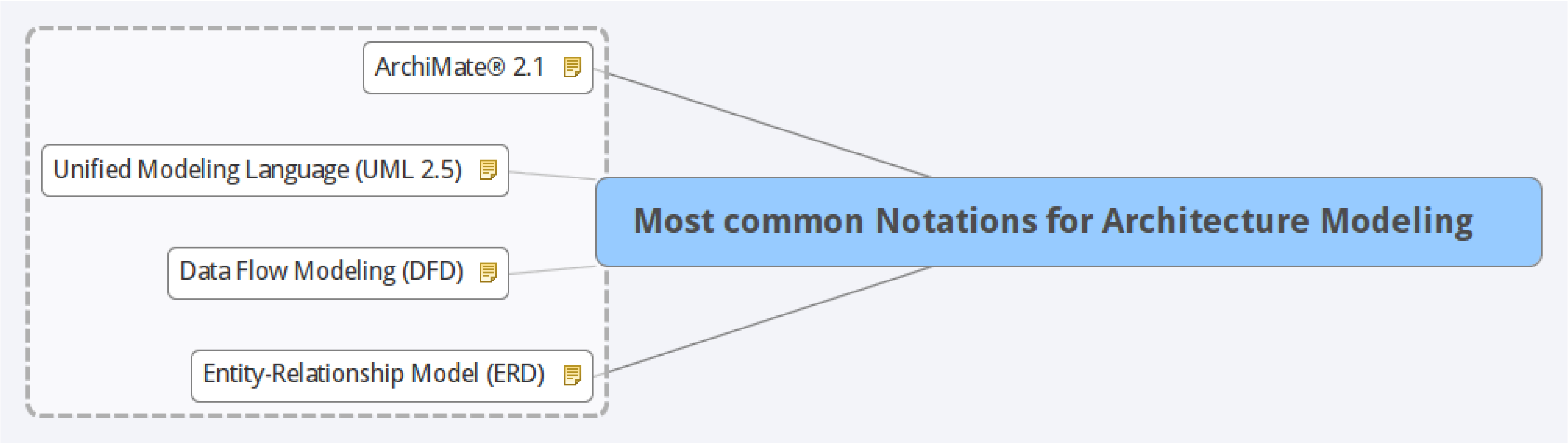
2003: TOGAF 8 ("Enterprise Edition") is published and updated regularly with new examples.

2009: TOGAF 9 includes: (1.) a new Content Meta-model that links the artifacts of TOGAF together, (2.) New Templates, (3.) Practices for Architecture scoping and segmentation, (4.) Practices for Business capability-based planning, (5.) Guidance on how to use TOGAF to develop Security Architectures and SOA.

2011: Maintenance release, TOGAF 9.1. is the current version of the Framework.

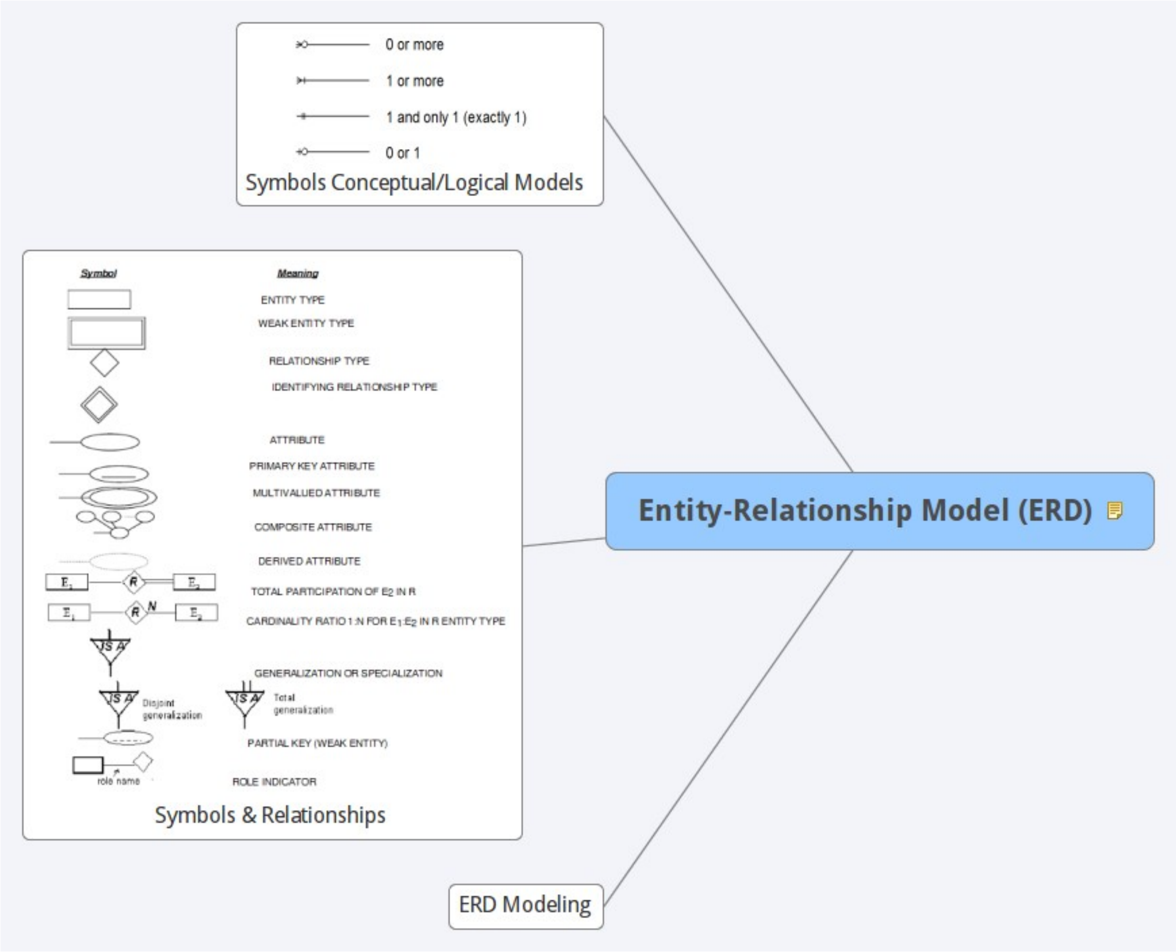
cc BY-NC-SA

Most common Notations for Architecture Modeling


ArchiMate® 2.1
Unified Modeling Language (UML 2.5)
Data Flow Modeling (DFD)
Entity-Relationship Model (ERD)
Most common Notations for Architecture Modeling

CC BY-NC-SA

Entity-Relationship Model (ERD)



Entity-Relationship Model (ERD)

0 or more

1 or more

1 and only 1 (exactly 1)

0 or 1

Symbols Conceptual/Logical Models

Symbol

Meaning

ENTITY TYPE

WEAK ENTITY TYPE

RELATIONSHIP TYPE

IDENTIFYING RELATIONSHIP TYPE

ATTRIBUTE

PRIMARY KEY ATTRIBUTE

MULTIVALUED ATTRIBUTE

COMPOSITE ATTRIBUTE

DERIVED ATTRIBUTE

TOTAL PARTICIPATION OF E_2 IN R

CARDINALITY RATIO 1:N FOR E_1:E_2 IN R ENTITY TYPE

GENERALIZATION OR SPECIALIZATION

PARTIAL KEY (WEAK ENTITY)

ROLE INDICATOR

Symbols & Relationships

ERD Modeling

(cc) BY-NC-SA

Entity-Relationship Model (ERD)

Entity relationship Diagramming is a popular, well standardized and supported graphical representation for Entities and their Relationships to each other, typically used organizing Data within relational Databases.

An Entity is a piece of data-an object or concept about which data is stored.

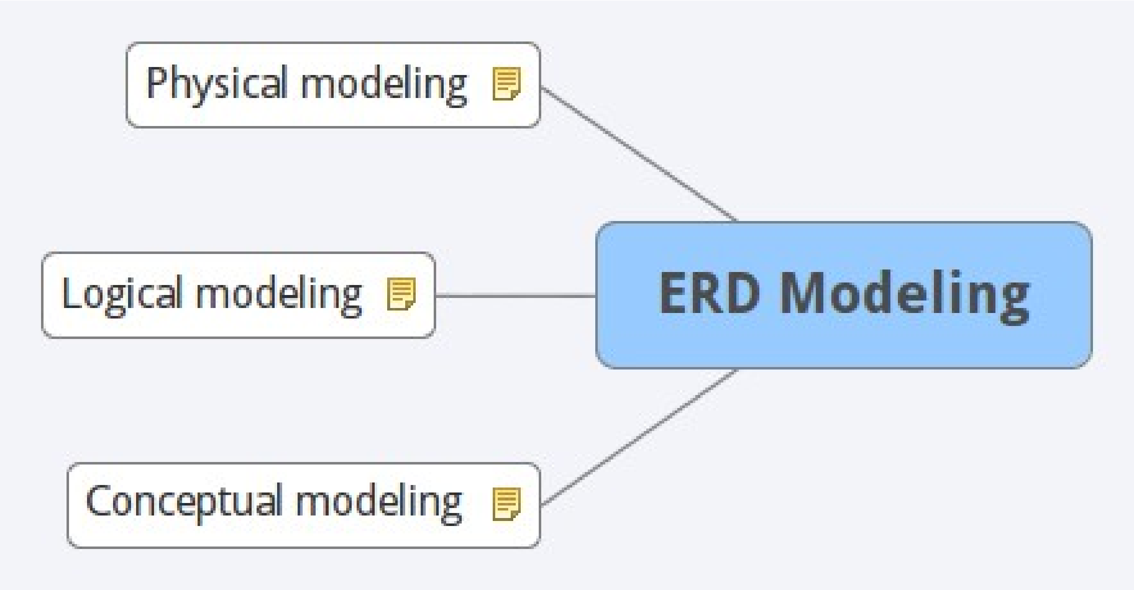
A Relationship illustrates an association between 2 Entities. Cardinality/Modality are the indicators of the business rules around a Relationship - i.e. how the data is shared between entities.

Cardinality refers to the maximum number of times an instance in one Entity can be associated with instances in the related entity.

Modality refers to the minimum number of times an instance in one entity can be associated with an instance in the related entity.

(cc) BY-NC-SA

ERD Modeling



(cc) BY-NC-SA

Conceptual modeling

A Conceptual Data Model is a high-level entity-relationship Model used to explore and explain basic business entities and the relationships between these entities.

It is used during preliminary requirements gathering to help participants understand the scope and purpose of the business opportunity being defined.

Detailed entity and attributes are not defined in a conceptual data model but general definitions of the business entities should include mention of the scope of information included in the entity.

A conceptual data model is not created for every Database and may result in one, many or no Databases.

(cc) BY-NC-SA

Logical modeling

A Logical Data Model is a structured Database design that defines the data requirements of the business in detail.

It includes all entities, attributes and relationships needed in the Database.

A logical data model is not DBMS specific.

Logical models are created for Operational/Repository type Databases.

A logical data model can generate a physical data model.

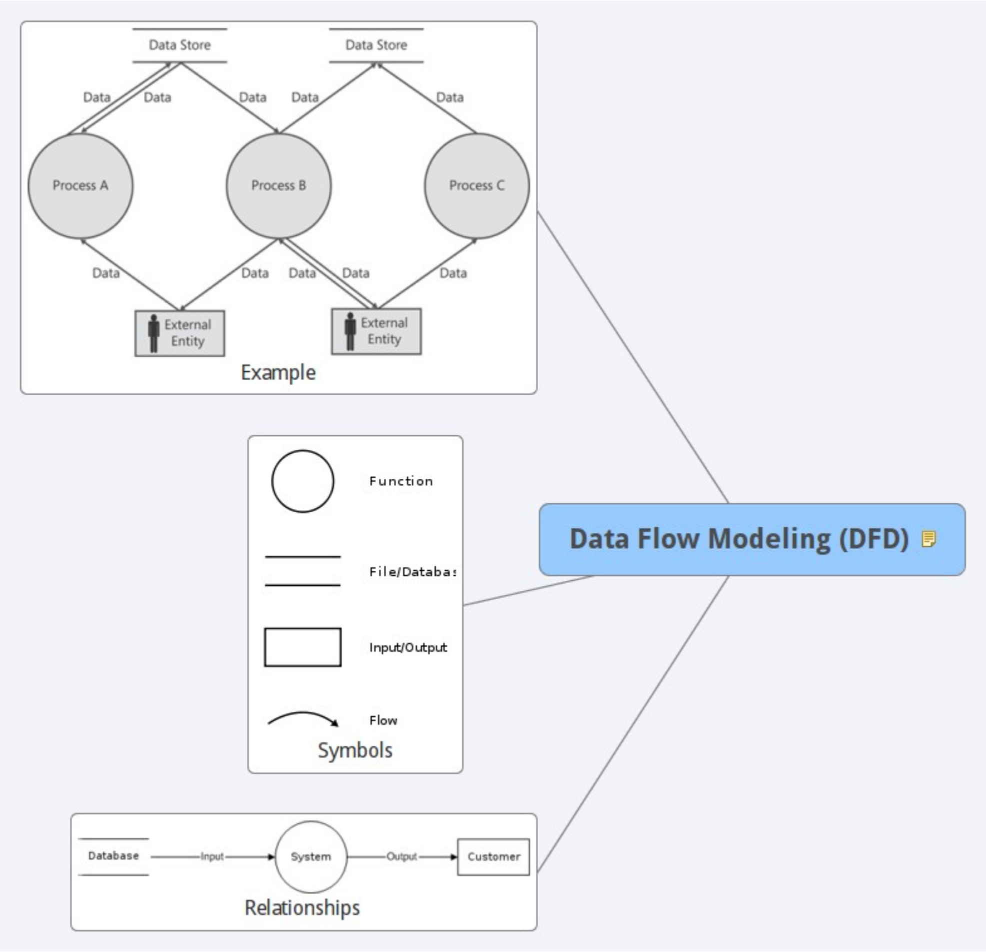
(cc) BY-NC-SA

Physical modeling

A Physical Data Model is the DBMS specific definition of the Tables, Columns and Relationships that define the structure of the Database.

(cc) BY-NC-SA

Data Flow Modeling (DFD)



(cc) BY-NC-SA

Data Flow Modeling (DFD)

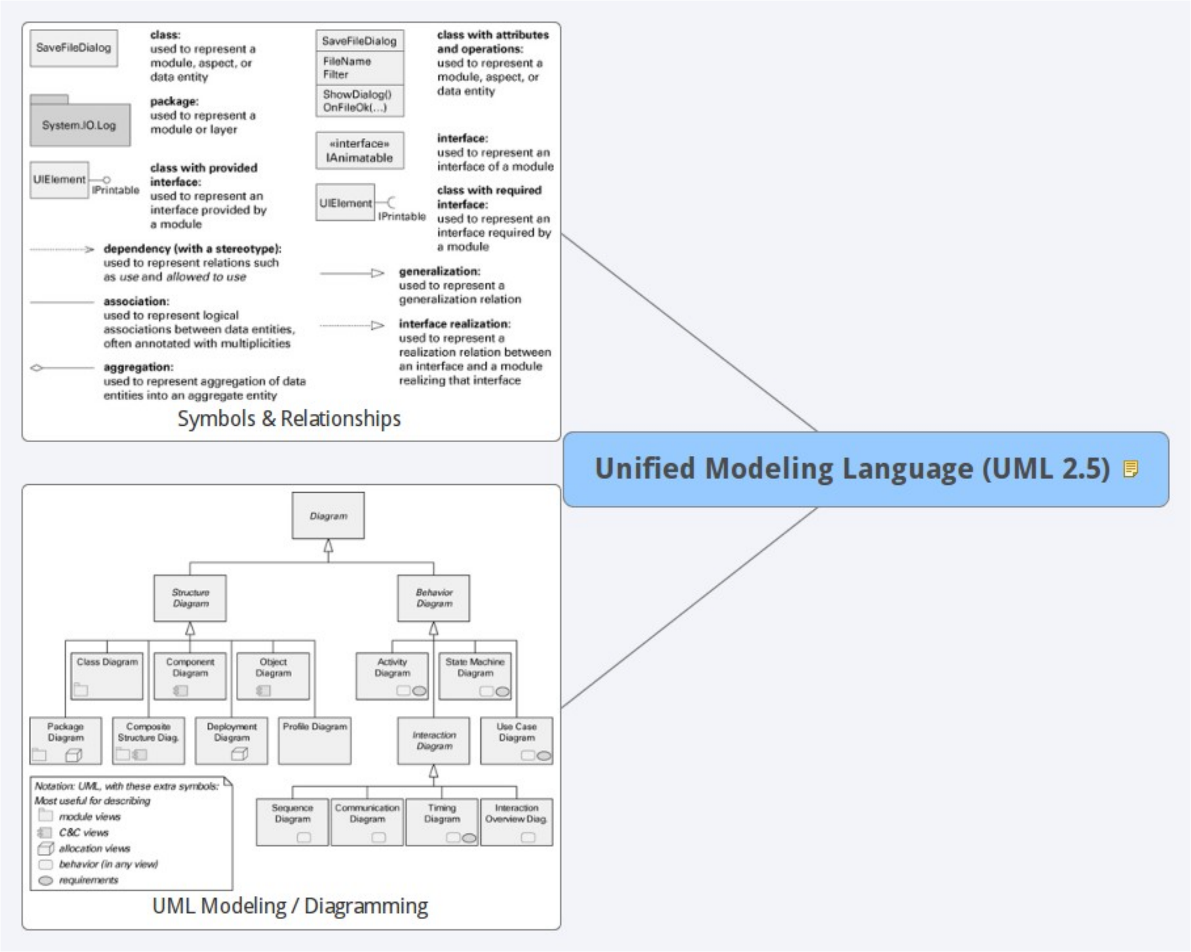
A data flow diagram (DFD) is a notation used for the visualization of data processing.

It is a graphical representation of the "flow" of data through an information system, modelling its process aspects.

It shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.

(cc) BY-NC-SA

Unified Modeling Language (UML 2.5)



class: used to represent a module, aspect, or data entity

package: used to represent a module or layer

class with provided interface: used to represent an interface provided by a module

dependency (with a stereotype): used to represent relations such as *use* and *allowed to use*

association: used to represent logical associations between data entities, often annotated with multiplicities

aggregation: used to represent aggregation of data entities into an aggregate entity

class with attributes and operations: used to represent a module, aspect, or data entity

interface: used to represent an interface of a module

class with required interface: used to represent an interface required by a module

generalization: used to represent a generalization relation

interface realization: used to represent a realization relation between an interface and a module realizing that interface

Notation: UML, with these extra symbols:
Most useful for describing
- *module views*
- *C&C views*
- *allocation views*
- *behavior (in any view)*
- *requirements*

cc BY-NC-SA

Unified Modeling Language (UML 2.5)

The UML standard is maintained by the Object Management Group (OMG).

It provides 14 types of diagrams divided into two categories: structure diagrams and behavior diagrams.

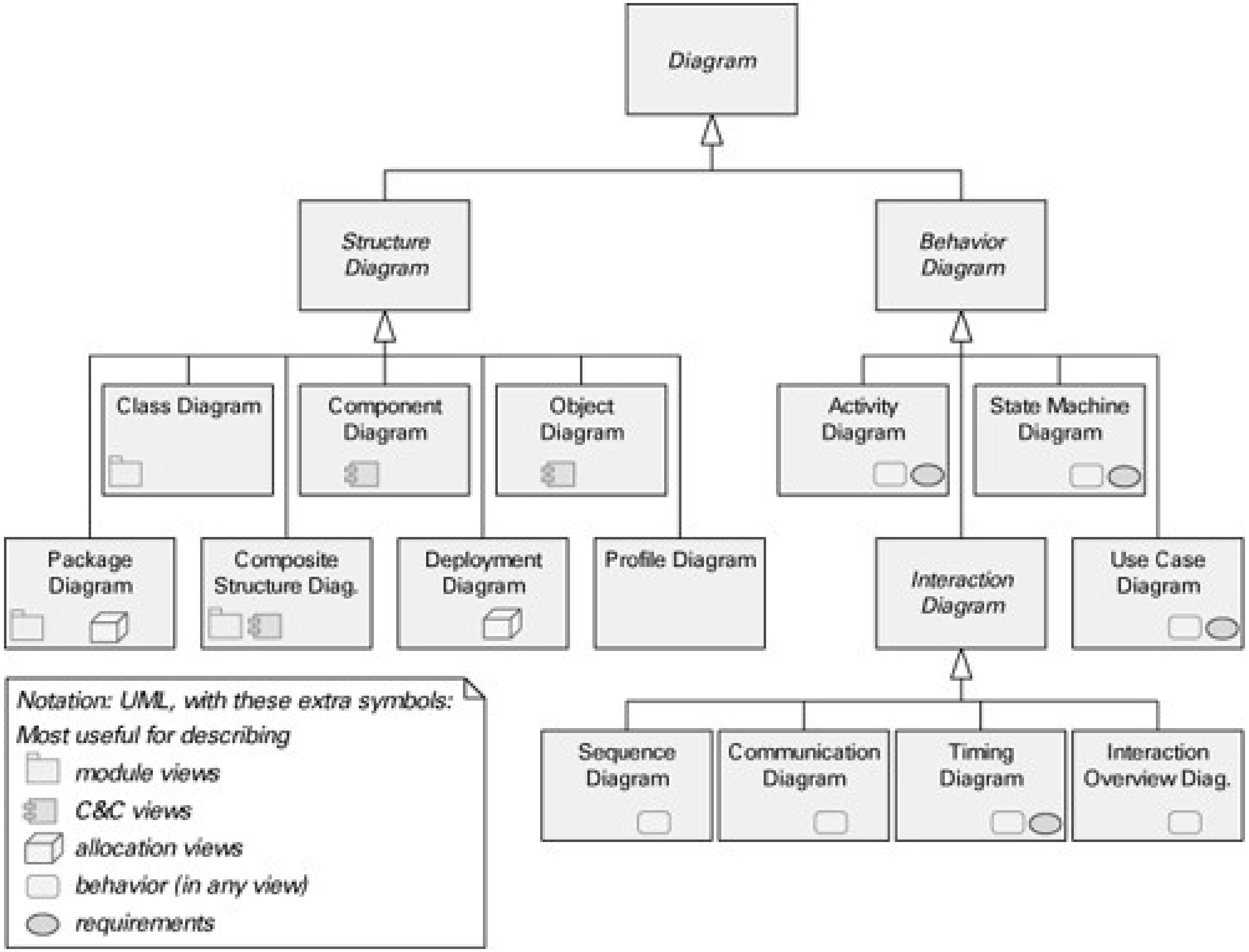
The meaning of any UML symbol can be specialized by using stereotypes.

A "Stereotype" is a domain-specific or technology-specific label shown within “angle brackets” that can be applied to existing UML elements to best describe an Architecture concept or relation.

UML is actively in practice today with strong tool support, especially in Manufacturing centric Industries (not so much in the non-manufacturing Enterprise).

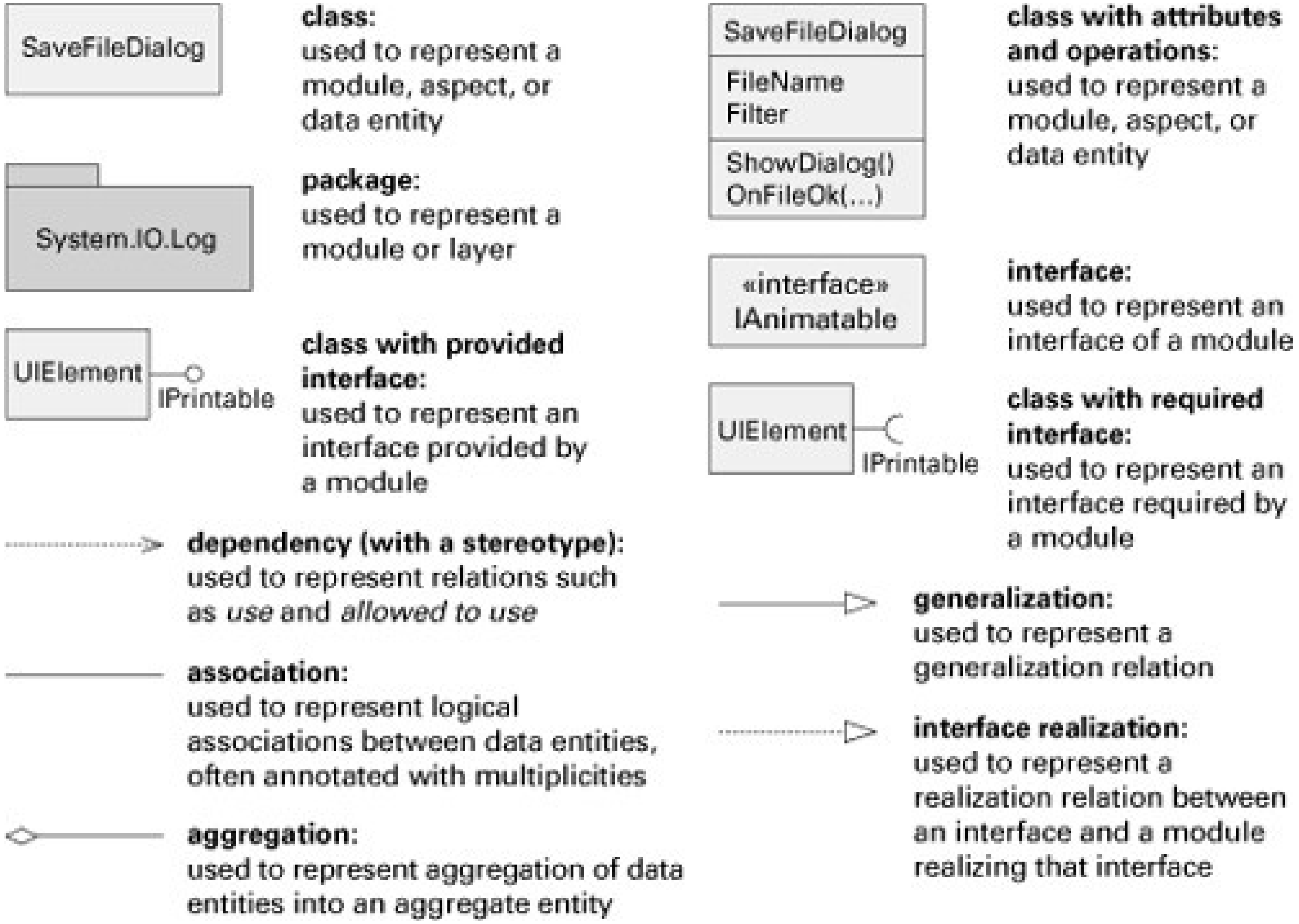
(cc) BY-NC-SA

UML Modeling / Diagramming



(cc) BY-NC-SA

Symbols & Relationships



class:
used to represent a module, aspect, or data entity

package:
used to represent a module or layer

class with provided interface:
used to represent an interface provided by a module

dependency (with a stereotype):
used to represent relations such as *use* and *allowed to use*

association:
used to represent logical associations between data entities, often annotated with multiplicities

aggregation:
used to represent aggregation of data entities into an aggregate entity

class with attributes and operations:
used to represent a module, aspect, or data entity

interface:
used to represent an interface of a module

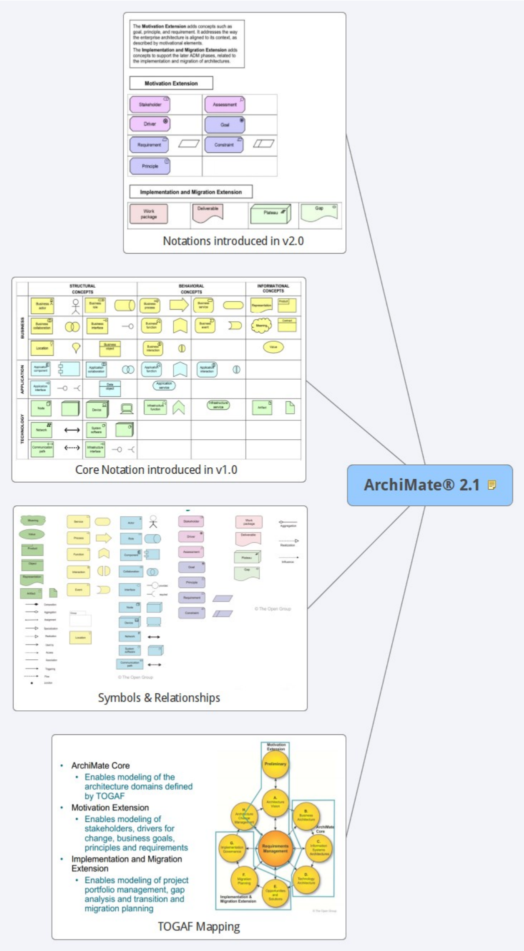
class with required interface:
used to represent an interface required by a module

generalization:
used to represent a generalization relation

interface realization:
used to represent a realization relation between an interface and a module realizing that interface

(cc) BY-NC-SA

ArchiMate® 2.1



The **Motivation Extension** adds concepts such as goal, principle, and requirement. It addresses the way the enterprise architecture is aligned to its context, as described by motivational elements.

The **Implementation and Migration Extension** adds concepts to support the later ADM phases, related to the implementation and migration of architectures.

- ArchiMate Core
  - Enables modeling of the architecture domains defined by TOGAF
- Motivation Extension
  - Enables modeling of stakeholders, drivers for change, business goals, principles and requirements
- Implementation and Migration Extension
  - Enables modeling of project portfolio management, gap analysis and transition and migration planning

(cc) BY-NC-SA

ArchiMate® 2.1

ArchiMate is an open and independent modeling language for EA, created by the Open Group and fully aligned with the TOGAF EA Framework.

It is a Notation for describing, analyzing and visualizing relationships amongst Architectural Domains as defined in TOGAF.

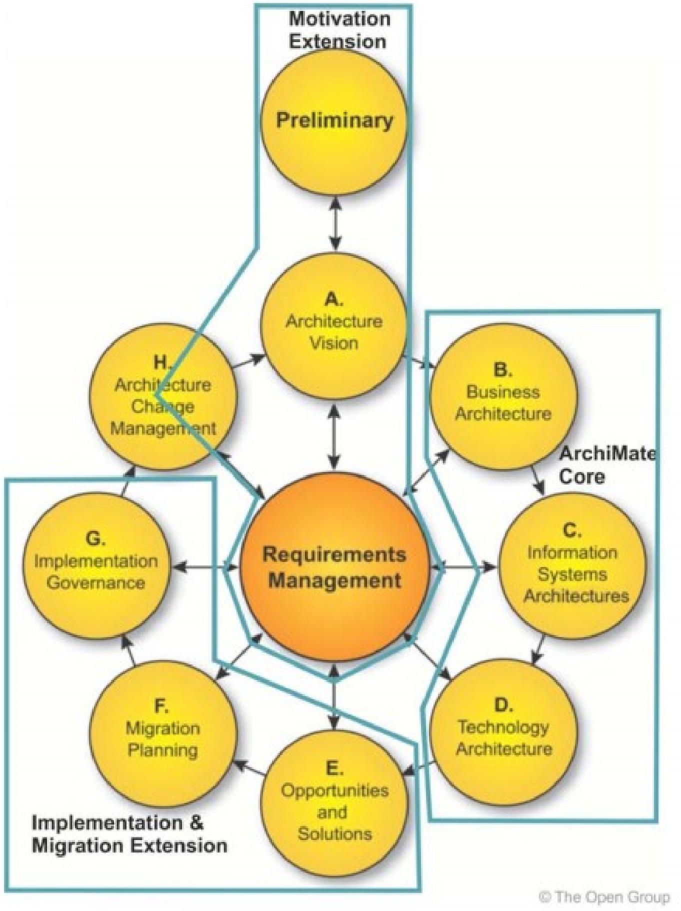
Each Domain is specified by a Meta-Model, constraining the Diagrams that can be created, and allowing consistency of notation and re-sue of Concept elements between Views.

ArchiMate also allows the connection of Models belonging to different "layers" (i.e. Business/Application/Data/Technology layers), hence helping an Architect to document View consistency.

(cc) BY-NC-SA

TOGAF Mapping

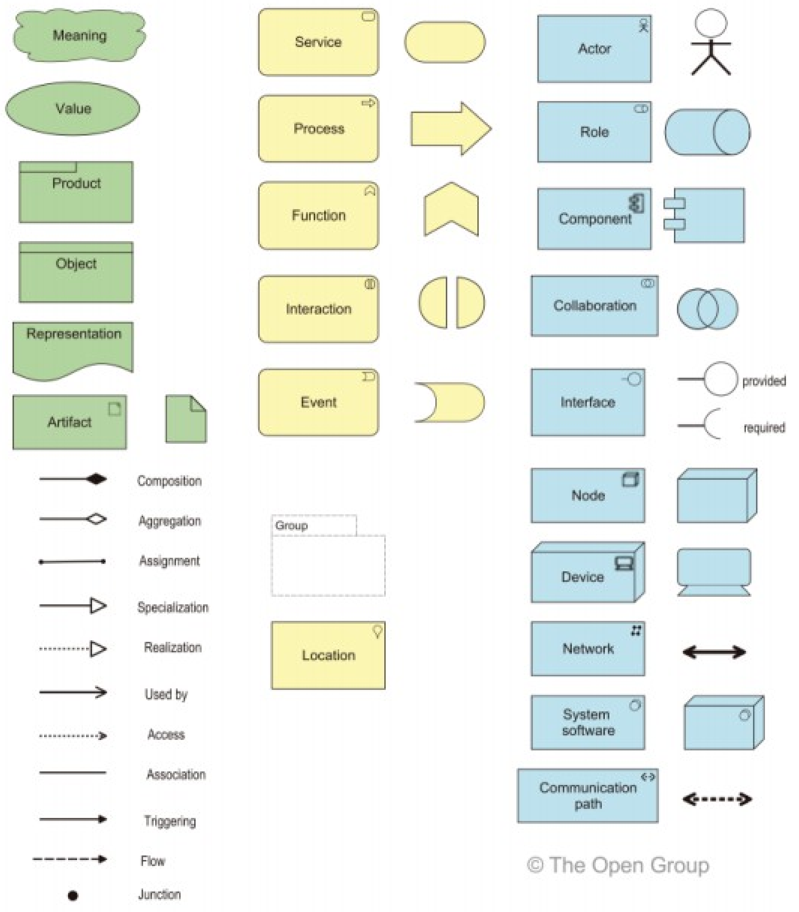
- ArchiMate Core
  - Enables modeling of the architecture domains defined by TOGAF
- Motivation Extension
  - Enables modeling of stakeholders, drivers for change, business goals, principles and requirements
- Implementation and Migration Extension
  - Enables modeling of project portfolio management, gap analysis and transition and migration planning

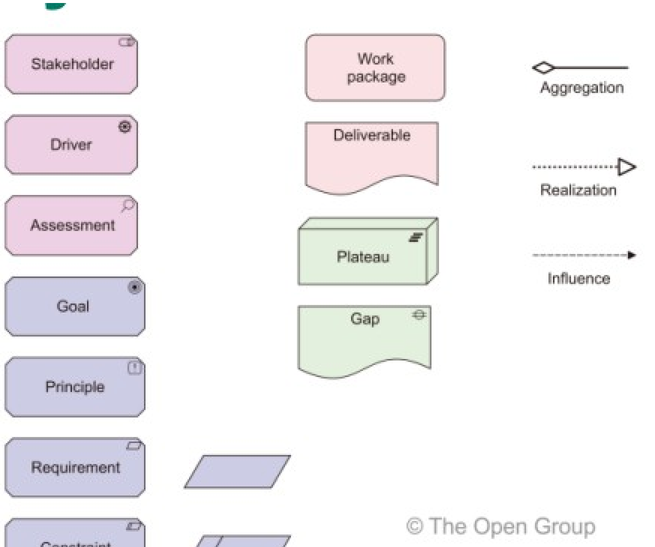

Motivation Extension
Preliminary
A. Architecture Vision
H. Architecture Change Management
B. Business Architecture
ArchiMate Core
Requirements Management
G. Implementation Governance
C. Information Systems Architectures
F. Migration Planning
D. Technology Architecture
E. Opportunities and Solutions
Implementation & Migration Extension

© The Open Group

cc BY-NC-SA

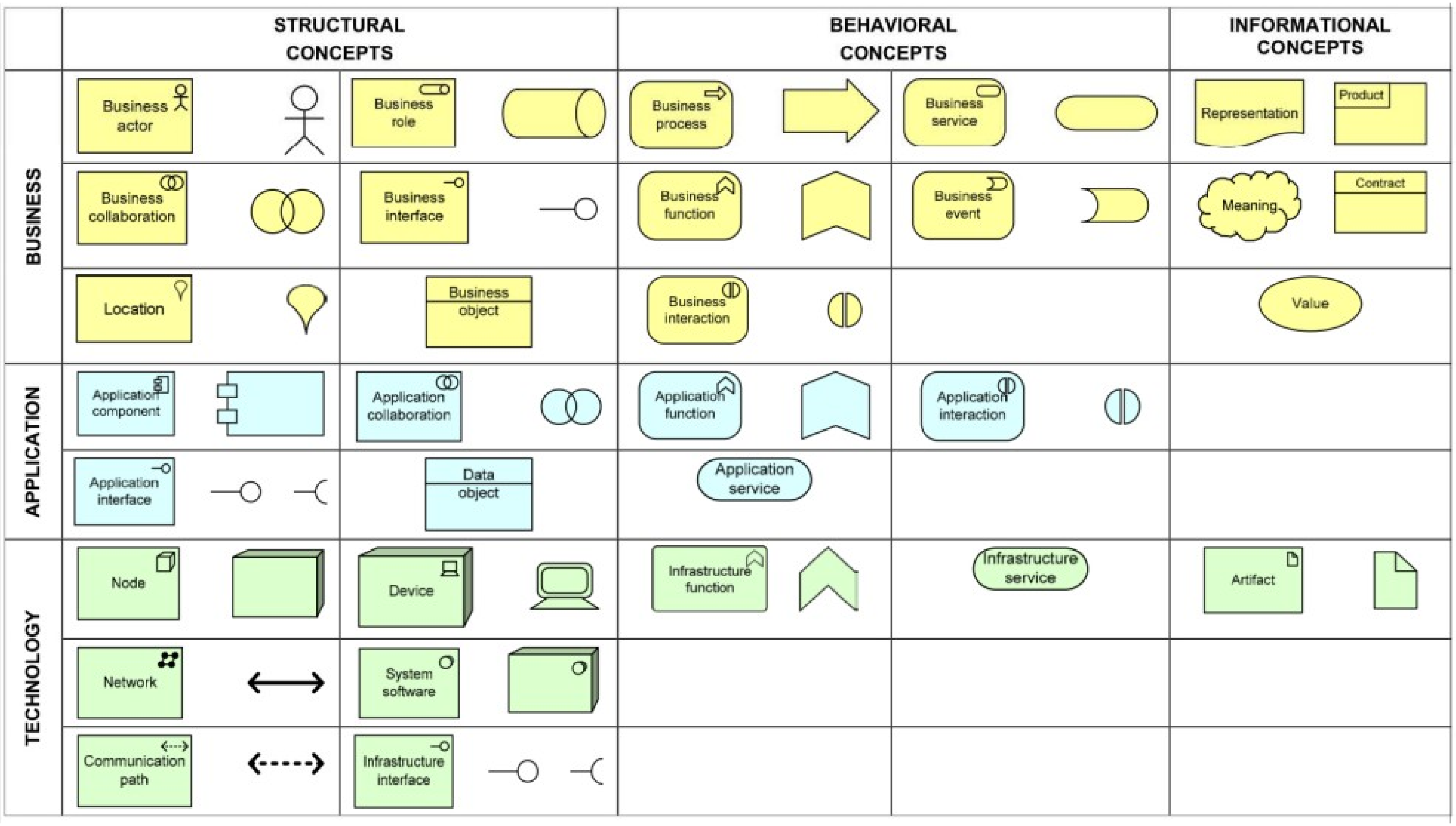
Symbols & Relationships





(cc) BY-NC-SA

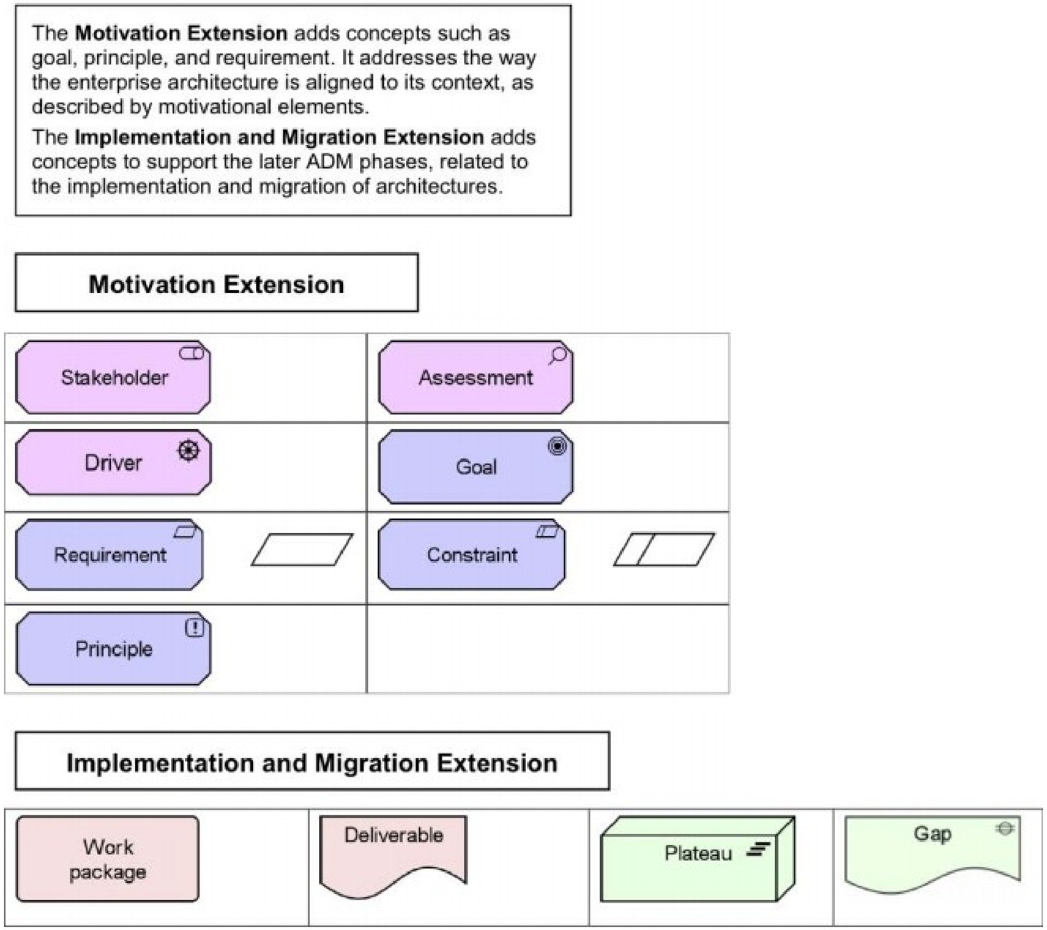
Core Notation introduced in v1.0



	STRUCTURAL CONCEPTS		BEHAVIORAL CONCEPTS		INFORMATIONAL CONCEPTS
BUSINESS	Business actor	Business role	Business process	Business service	Representation Product
	Business collaboration	Business interface	Business function	Business event	Meaning Contract
	Location	Business object	Business interaction		Value
APPLICATION	Application component	Application collaboration	Application function	Application interaction	
	Application interface	Data object	Application service		
TECHNOLOGY	Node	Device	Infrastructure function	Infrastructure service	Artifact
	Network	System software			
	Communication path	Infrastructure interface			

cc BY-NC-SA

Notations introduced in v2.0



The **Motivation Extension** adds concepts such as goal, principle, and requirement. It addresses the way the enterprise architecture is aligned to its context, as described by motivational elements.

The **Implementation and Migration Extension** adds concepts to support the later ADM phases, related to the implementation and migration of architectures.

(cc) BY-NC-SA