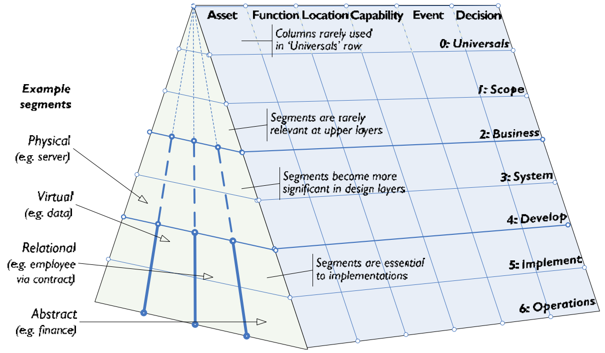
A framework for whole-of-enterprise architecture

The framework provides a means to categorise and cross-reference the various business-entities identified in the architecture, and determines the business meaning of each entity. The root-level of the framework *must* always cover the whole scope of the enterprise, not just its IT.

The framework described here is an extension of the well-known Zachman taxonomy:

- clarifies the business function of each of the six columns
- adds an extra 'row-0' at the top of the frame, to hold enterprise 'Universals'
- adds an extra 'segments' dimension, to distinguish between physical items, virtual items, people-relationships-as-items; or functions done by machines, by software, by people; and so on.


Asset
Function
Location
Capability
Event
Decision
Columns rarely used in 'Universals' row
0: Universals
1: Scope
Segments are rarely relevant at upper layers
2: Business
3: System
Segments become more significant in design layers
4: Develop
5: Implement
Segments are essential to implementations
6: Operations
Example segments
Physical (e.g. server)
Virtual (e.g. data)
Relational (e.g. employee via contract)
Abstract (e.g. finance)

Rows, columns and segments

Primitives and composites

The framework describes the 'what' of the architecture, in terms of *primitives* and *composites*:

- primitives are the 'atoms' or root-elements of the architecture
- composites are the 'molecules' or architectural solution-components constructed from any required combination of simpler entities

Composites may often be layered, much as a complex molecule such as DNA is itself structured from simpler molecules. We must also be able to resolve every composite to its root-primitives, because:

Primitive models are architecture. Composite models are implementations.

Every composite is a pre-packaged 'solution', to guide *solution*-architecture, the processes that occur *after* the core assessment work for *enterprise*-architecture. If we cannot resolve all the way to root-primitives, we will be stuck with existing solutions, which may not fit a changed context. If we don't have the right set of root-primitives, we'll be unable to *see* new solutions, or have any way to rethink what's going on in our enterprise. Flexibility depends on the precision of the metamodel. Most of the time we work well above true primitives, but in planning for disaster-recovery, or for major legislative change we really *do* need to get right down to the roots and rethink things from scratch.

Each cell in the framework represents a class of primitives, whilst composites straddle horizontally across multiple cells of the framework. There are no vertical composites: instead, vertical structures such as logical-to-physical data-maps are more accurately trails of derivation or implementation.

Moving down the framework layers also moves toward architectural 'completion' and actual *use*. At row-5 and -6, a solution-design cannot work if any column is missing. But if it's complete, change is not possible: abstraction allows substitution, replacement, re-design. Hence an important principle:

Composites are usable to the extent that they're architecturally complete;
composites are *re*-usable to the extent that they're architecturally *in*complete

Also related to incompleteness is 'bindedness' – the extent to which a particular item must be included or applied within a solution, as defined, for example, in a Technical Reference Model.

Framework – layers

For the vertical dimension of the framework, we partition scope in terms of timescale – a set of seven distinct layers or perspectives, from unchanging constants, to items which change moment by moment. Each row adds another concern or attribute, as follows:

- *Row 0*: '**Universals**' – in principle, should never change: core constants to which everything should align – identifies the overall region of interest and the key points of connection shared with enterprise partners and other stakeholders; added for compatibility with ISO-9000 etc
- *Row 1*: '**Scope**' (*Zachman*: 'Planner') – adds possibility of change, if usually slowly: core entities in each category, in list form, *without* relationships - the key 'items of interest' for the enterprise
- *Row 2*: '**Business**' (*Zachman*: 'Owner') – adds relationships and dependencies between entities: core entities described in summary-form for business-metrics, including relationships between entities both of the same type ('primitives') and of different types ('composites')
- *Row 3*: '**System**' (*Zachman*: 'Designer') – adds attributes to abstract 'logical' entities: entities expanded out into implementation-*independent* designs - includes descriptive attributes
- *Row 4*: '**Develop**' (*Zachman*: 'Builder') – adds details for real-world 'physical' design: entities and attributes expanded out into implementation-*dependent* designs, including additional patterns such as cross-reference tables for 'many-to-many' data-relationships
- *Row 5*: '**Implement**' (*Zachman*: 'Sub-contractor' or 'Out of Scope') – adds details of intended future deployment: implementation of designs into actual software, actual business-processes, work-instructions, hardware, networks etc
- *Row 6*: '**Operations**' (*Zachman*: implied but not described) – adds details of actual usage: specific instances of entities, processes etc, as created, modified, and acted on in real-time operations

Each row represents a different category of responsibility or stakeholder, such as senior management responsible for row-0 universals, or strategists at row-1 and -2, architects and solution-designers at row-3 and -4, and line managers and front-line staff at row-5 and -6.

Framework – columns and segments

Below the core-constant 'Universals', the framework splits horizontally into columns for six distinct major categories of primitives – approximating to what, how, where, who, when and why. In the lower layers of the framework, we also need to split the columns themselves by context into distinct segments or sub-categories – typically related to people, information, things and essential abstracts.

The ***columns*** or content-types and primitive-types:

- *What*: **assets** of any kind – physical objects, data, links to people, morale, finances, etc
- *How*: **functions** – activities or services to create change, described independently from the agent (machine, software, person etc) that carries out that activity
- *Where*: **locations** – in physical space (geography etc), virtual space (IP nodes, http addresses etc), relational space (social networks etc), time and suchlike
- *Who*: **capabilities** clustered as **roles** or 'actors' – may be human, machine, software application, etc, and individual or collective
- *When*: **events** and relationships between those events – may be in time, or physical, virtual, human, business-rule trigger or other event
- *Why*: **reasons**, decisions, constraints and other tests which trigger or validate the condition for the 'reason' and the like, as in strategy, policy, business-requirements, business-rules, regulations etc.

The ***segments*** or sub-categories within the columns could be cut multiple ways, but typically:

- **physical**: tangible objects (What), mechanical processes (How), physical or temporal locations (Where), physical events (When); also align to ***rule-based*** skills (Who) and decisions (Why)
- **virtual**: intangible objects such as data (What), software processes (How), logical locations (Where), data-driven events (When); also align to ***analytic*** skills (Who) and decisions (Why)
- **relational**: links to people (What), manual processes (How), social/relational locations (Where), human events (When); also align to ***heuristic*** skills (Who) and decisions (Why)
- **aspirational**: principles and values (What), value-webs and dependencies (Where), business-rules (When); also align with ***principle-based*** skills (Who) and decisions (Why)
- **abstract**: additional uncategorised segments such as financial (What, How), energy (What) etc

Relationships may exist in columns between primitives of the same segment, or in different segments.

At the row-5 / Implement level, it should be possible to describe *everything* in a single sentence-structure, as a distinct set of primitives, and as a composite straddling across every column:

"with «*asset*» do «*function*» at «*location*» using «*capability*» on «*event*» because «*reason*»"

At the row-6/Operations level, the sentence is usually in the past tense, but still essentially the same:

"«*function*» was done with «*asset*» by «*capability*» at «*location*» on «*event*» because «*reason*»"

This enables architectural redesign, anchoring trails of relationship between items or layers, to resolve *business*-concerns such as strategic analysis, failure-impact analysis and complex 'pain-points'.

Assets – 'what?' (Zachman: *data*, or more generally as '*entity / relationship*')

The assets are countable and measurable 'things', with the relationships between assets of the same type, as primitives, and different types, as composites. Column segments include:

- *physical assets*: servers, routers, widgets, paper forms, physical objects of all kinds
  - model types: parts-breakdown, bill of materials, etc
- *virtual assets*: data, metadata, messages, models
  - model types: data-model, metadata-schema, model-definitions etc
- *relational assets*: links to people – customers, clients, employees and other stakeholders
  - model types: business-relationships and relationship-types
- *aspirational assets*: morale, values, commitment, drive
  - model types: architecture 'Universals' (row-zero), customer / employee survey-designs etc
- *abstract assets*: financials, valuations ('goodwill'), energy-resources (electricity, water, network-traffic) etc
  - model types: financial models, derivatives, valuation schemas etc

The distinctions between the segments are derived from standard classification-schemes for property types – for example 'alienable' (physical) versus 'non-alienable' (virtual).

Functions – 'how?' (Zachman: 'process / input-output')

The primitive is *function* in the mathematical sense: a change between input and output, as *a=func(x,y)*. Since functions act on things - How acts on What - the segments should be as for the What column:

- *physical*: transform physical objects
- *virtual*: transform data and other virtual information
- *relational*: transform business-relationships - for example, to close a sale
- *abstract*: transform financials - for example, financial derivatives

In principle there are also functions that operate on aspirational assets such as morale and values.

Locations – 'where?' (Zachman: 'node / link'; "the locations relevant to the organisation")

The terms 'node', 'location' and 'link' must expand to cover the full range of 'What'-type segments:

- *physical*: geographic locations, building/room-numbers and other physical locations
  - model-types: logistics maps, schematics, map-coordinates, etc
- *virtual*: network locations, IP addresses, web or Java-style code-addresses, telephone numbers
  - model-types: network-maps, network-coordinates, number-allocation maps, file-hierarchies etc
- *relational*: relative locations for people
  - model-types: social-network maps, reporting-relationship matrices, etc
- *aspirational* and/or *abstract*: value-webs, dependencies
  - model-types: Porter value-chains, dependency-trails, audit-trails, etc
- *abstract*: temporal locations, etc
  - model-types: timelines, project timescales, etc

Note that time is Where, not When': When-events may happen *in* time, but time *itself* is not an event.

The 'link' relationships of the location-primitive create the same kind of expanding decompositions as for What, without the primitive-to-composite complications of the How column.

Capabilities – 'who?' (Zachman: '*people*' or 'agent'; 'relationships between people and work')

The appropriate primitive here is *capability*, usually clustered into sets as ***roles*** and their *responsibilities*. The role is always abstract: for real use, it must be linked with a suitable 'What' into an 'agent' composite. Recommended segments are skill-levels, as implied in the Cynefin complexity model:

- *rule-based* (Cynefin 'known' domain; aligns with *physical* segments): no choice or judgement permitted, hence little to no true skill (i.e. training only); real-time or near-real-time only, and strictly causal; may be implemented by machines, IT or people
- *analytic* (Cynefin 'knowable' domain; aligns with *virtual* segments): some judgement required, with analytic choices amongst various paths, which may include disconnects in time; modelled as cause-effect relationships; cannot implement by machines without IT-based or human support
- *heuristic* (Cynefin 'complex' domain; aligns with *relational* segments): true skills, judgement always required to assess patterns in contexts of high uncertainty, limited statistical probability, complex delay-loops and/or unknown factors; apparent cause-effect patterns identified retrospectively; cannot be implemented by machines or conventional IT-based systems without human support
- *principle-based* (Cynefin 'chaotic' domain; aligns with *aspirational* and/or *abstract* segments): very high skill-levels, extreme and inherent uncertainty in real-time or near-real-time; no direct cause-effect patterns identifiable (e.g. 'market of one'); can only be implemented by humans

To summarise, the appropriate primitive for the Who column is not 'people', but generic *capability*.

Events – 'when?' (Zachman: '*Time / cycle*')

The primitive here is the generic '*event*', and relationships between these events. It's simplest to use the same segments as for the 'What' column:

- *physical events*: includes 'disaster' incidents - fire, flood etc - and any kind of physical trigger-event
- *virtual events*: includes messages and other data-triggers
- *relational events*: includes people-based trigger-events such as meetings, phone-contacts, sign-offs
- *aspirational events*: includes business-rules (because these focus on meaning and purpose)
- *abstract events*: includes time-based triggers, business-cycles and the like

Relationships between events will often be nested and hierarchical: for example, cycles may contain smaller cycles, and so on ad infinitum. Other relationships may be chained, as in a project Gantt chart.

Reasons – 'why?' (Zachman: '*motivation*', more generally as 'ends / means'; also 'business rules')

The appropriate primitive here is '*decision*', linked via derivation-trails to '*the* decision', the enterprise Vision. Decisions and 'reasonings' underpin motivation, so Zachman's description for the column is still be valid. Business rules, requirements, constraints, policies, analyses and other decisions fo strategy and tactics are the *reasons* we do anything in the enterprise, and should be documented in this column. The Cynefin set is the most useful cut for segments, indicating 'bindedness' of decision:

- *rule-based* (Cynefin 'known' domain): 'law', mandatory, no flexibility – must always link to source(s) that justify the rigidity of the decision, such as analytics, external legislation, etc
- *analytic* (Cynefin 'knowable' domain): 'best-practice' decision on multiple factors or transforms – link to the factors (What), transforms (How) etc on which it is based, and to sources for standards
- *heuristic* (Cynefin 'complex' domain): contextual guidelines – links to other decisions or items identifying the context, and to roles or persons responsible for maintenance of the guidelines
- *principle-based* (Cynefin 'chaotic' domain): decision derived from high level of skill – must link to guiding principles and to person or group (e.g. committee) responsible for the decision

This matches ISO-9000: rule-based *work-instructions* derive from analytic or best-practice *procedures* derived from heuristic *policies* or guidelines, derived in turn from principles indicated by the *vision*. Relationships between reasons include '‹expands on›', 'conflicts with', and also '‹implements›'.

Resources

For more details on this adaptation of Zachman, and use of the TOGAF ADM for whole-of-enterprise architecture, see Tom Graves, *Bridging the Silos: enterprise-architecture for IT-architects* (Tetradian Books, 2008), ISBN 978-1-906681-02-9; publisher-details at www.tetradianbooks.com/2008/04/silos/

For details on the original Zachman Framework specification, see www.zifa.com