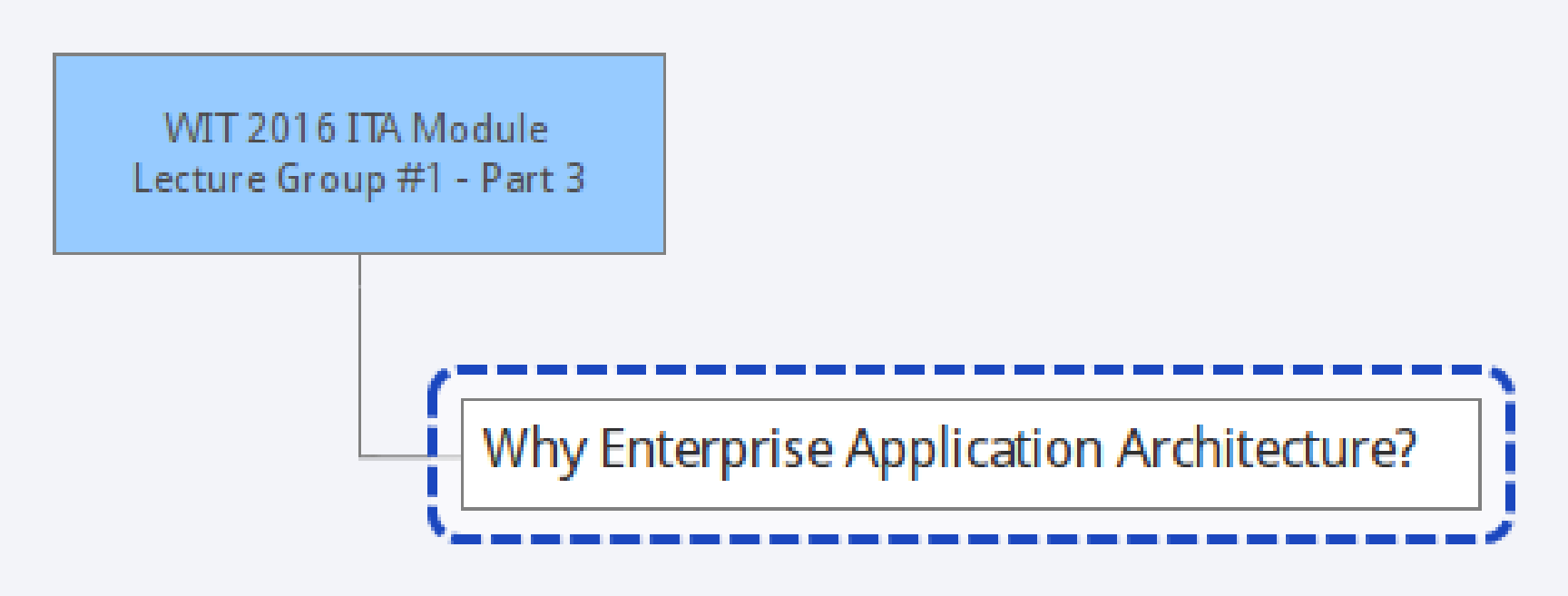
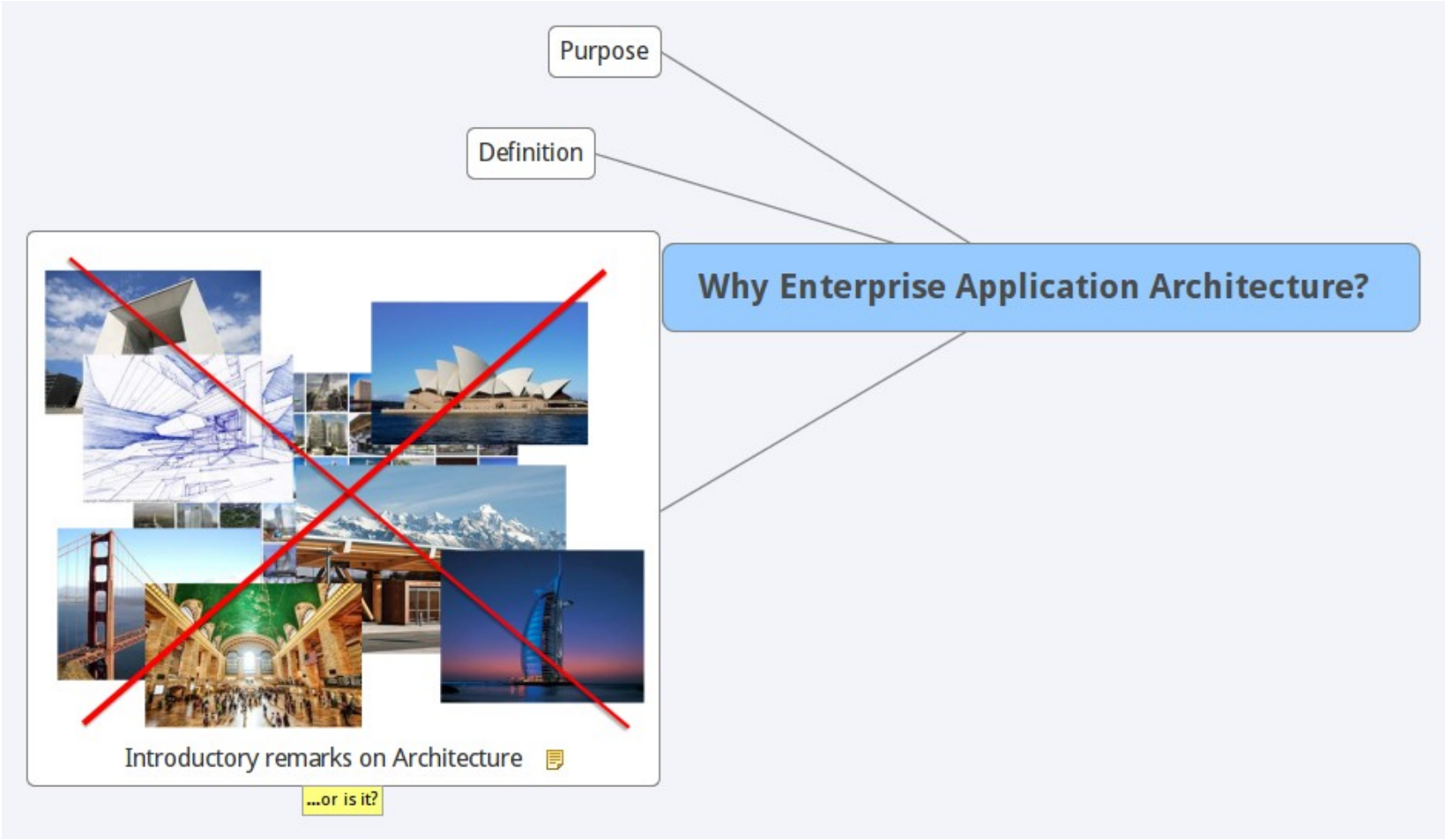
WIT 2016 ITA Module

Lecture Group #1 - Part 3

Lecture Group #1 - Part 3



Why Enterprise Application Architecture?



Introductory remarks on Architecture

"Architecture" is one of those words that has long migrated into our common language.

Its manifestations are visible all around us and are an integral part of our daily lives, in both Physical and Digital ways.

We think we understand each other when we talk about Architecture.

And indeed, precise enough definitions of Architecture are captured in dictionaries and encyclopedias.

But most refer to the "classic" definition of the term from the building construction industry.

As a result Architecture typically refers to monuments or structures, visual aesthetics or features smartly designed.

Few of us think of Architecture as the complex & modern field of Design & Engineering it is today.

Need for Architecture in the Digital Age

A firm is kept in business by a core set of Business Processes and Systems performing thousands (...) of transactions, daily.

The sound structure of systems & processes supporting these transactions can help or hinder the efforts of a Company to compete.

Enterprise Architecture is the explicit design of such systems & processes - fulfilling the Operating Model (OM) of the Enterprise.

It has a vital role in enabling (or constraining!): (a.) Business performance, (b.) Decision making, (c.) Strategy execution.

To generate a distinct competitive advantage, modern organizations can no longer separate discussions about an IT Project and a Business Initiative.

Stovepipes, or "islands" still exist in many Firms however.

Architecture aims to ensure that the Enterprise gets value back from investments in Technology and Information assets, across its organization.

Module Guiding Statements

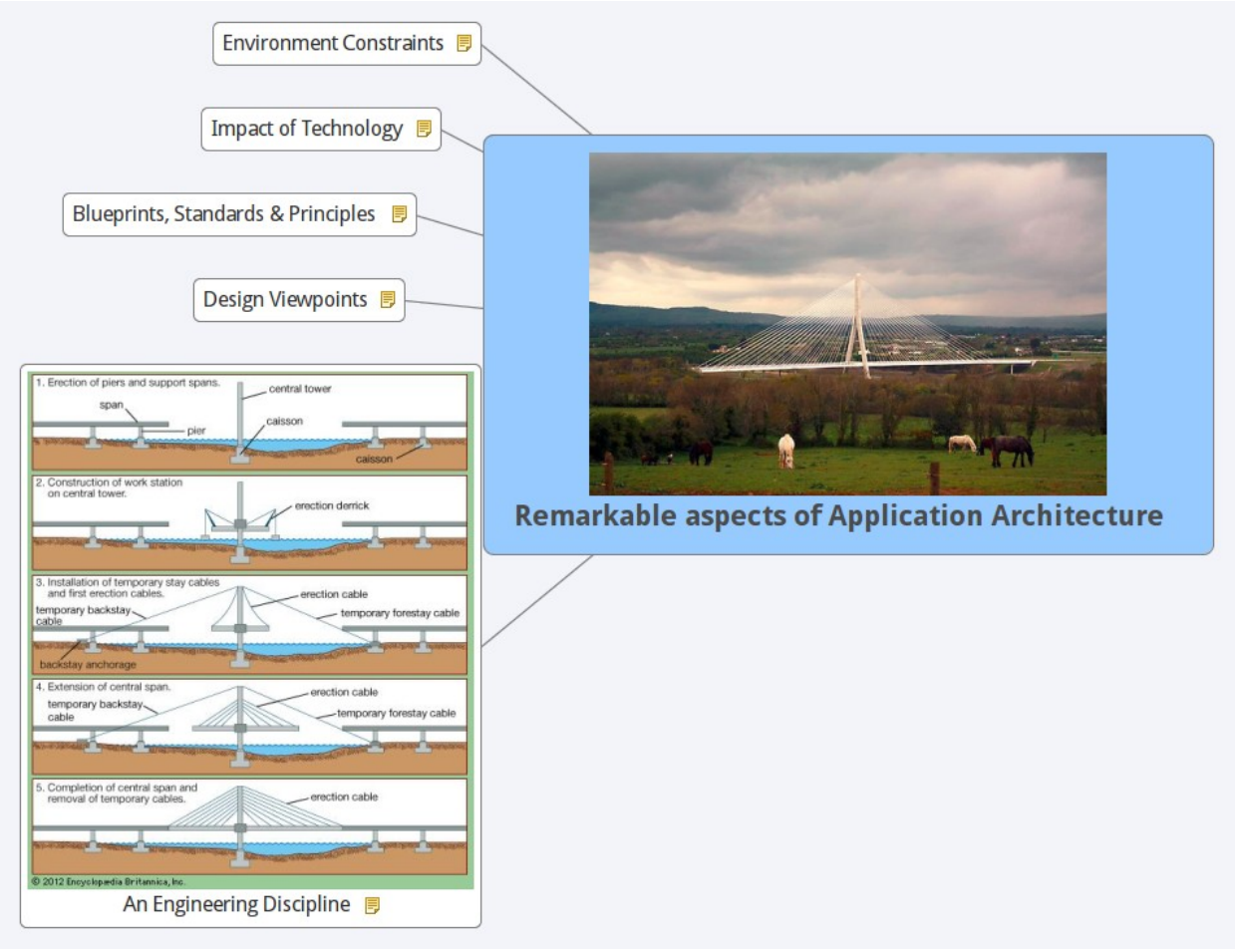
IN this module, Architecture is an ENGINEERING discipline: Enterprise Application Architecture.

It aims to engineer the Enterprise as a SYSTEM - i.e. the Digital Nervous System of the Enterprise.

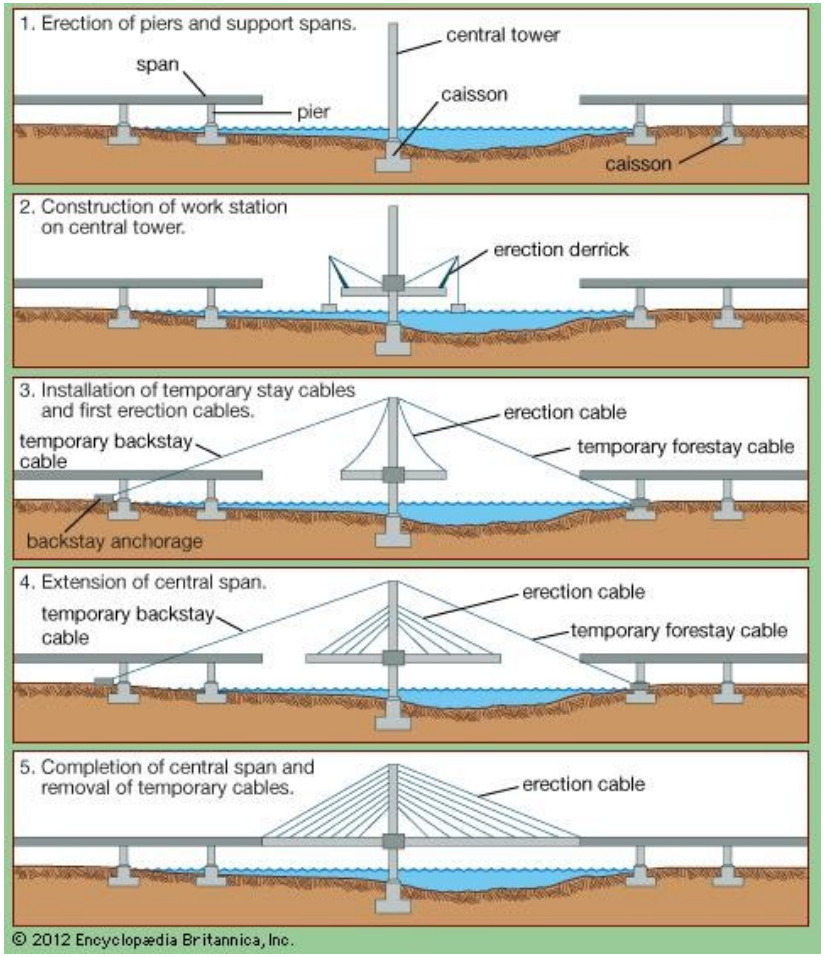
It realizes the VISION for an Enterprise as defined by its Executive Management, through the use of Information Technologies & Information Management Systems.

As such it is about designing Architectural SOLUTIONS for the BUSINESS Enterprise (or any organization in general - i.e. Government or Non-profit organizations).

Remarkable aspects of Application Architecture



An Engineering Discipline



An Engineering Discipline

Architecture is a Planning and Engineering story before being a Manufacturing story.

The Solution design is defined as a Plan (planning activities).

The Solution design is represented as a Model (modeling activities).

The Solution design helps the sequencing and execution of the construction (material selection and procurement, manufacturing, many more).

...and Architectural oversight spans the ENTIRE life of the resulting construction.

Design Viewpoints

Complexity comes from the intertwined and interdependent strings of "trade": software, data, technology.

Each trade holds a deep and ever increasing subject matter of expertise.

Engineering Viewpoints help to breakdown this complexity; outline, capture and formalize relationships and dependencies between each thread.

Blueprints, Standards & Principles

In the building industry Nowadays all building construction materials are codified. Material properties (ex. resistance) are known.

Building a suspension bridge or a cable-strayed bridge remains a significant undertaking, but blueprints (i.e. styles) are made adjusted to local topologies.

The exact same pertains to Solution Architecture in the enterprise as we will uncover in the Module.

Impact of Technology

Within the building industry, new indoor and outdoor materials presenting new properties, push the limits of what can be built.

With every new Technology introduced comes new possibilities.

Ambitious architectures, yesterday not economically viable, are today enabled by new manufacturing methods, lowering down the cost of material construction materials.

The exact same pertains to Solution Architecture in the enterprise.

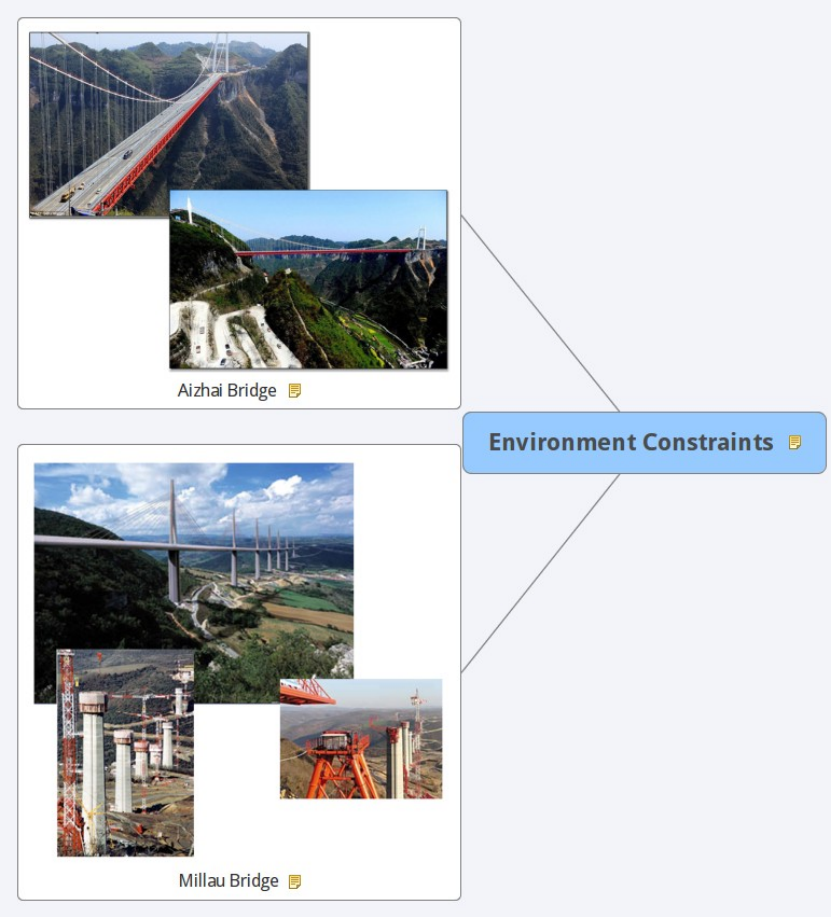
However it isn't all about the "cost of entry". It is also about managing the Enterprise's technical debt.

Environment Constraints

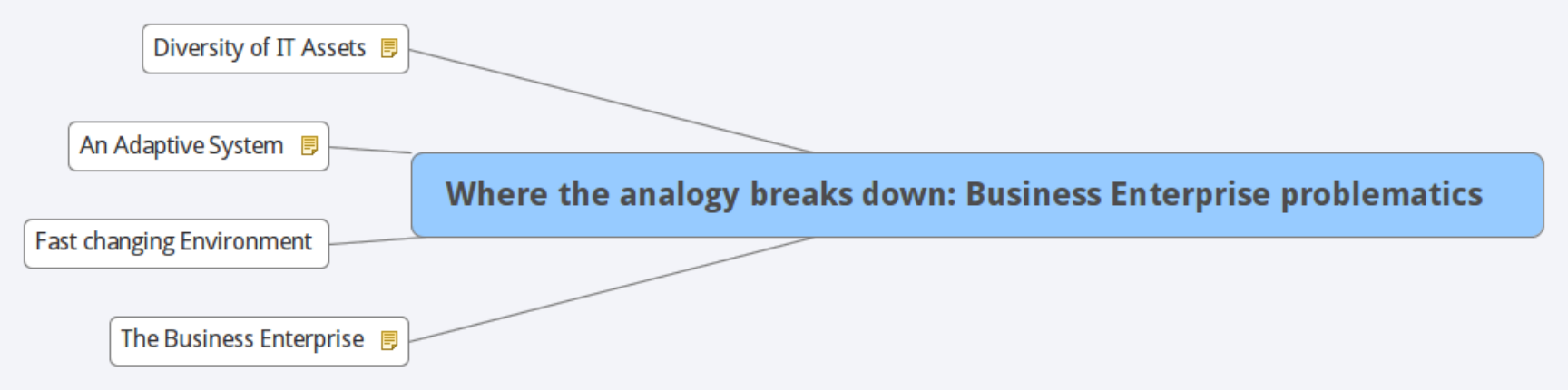
Environmental constraints generate Architectural requirements that are not functional in nature.

When applied on the envisioned design, such constraints forces Architectural trade-offs.

When applied on the envisioned design, such constraints inspire creative solutions.



Where the analogy breaks down: Business Enterprise problematics



The Business Enterprise

The Enterprise is made of People, its workforce and main asset.

The Enterprise is a People story before being a Product or Money story.

An Enterprise is as good as its People are.

Life-span of Enterprise last (ideally) many years.

People with expertise come and go... Creating an undesirable side-effect: short-term memory syndrome.

Think of it as forgetting about what you did (and how you did it) every week, week after week.

How do you sustain the long-term memory of your Company?

Fast changing Environment

The catch

Fast changing Environment

City Planning

City Planners have Building Codes to define where you can build and what Building Materials may be used for construction of each type of building.

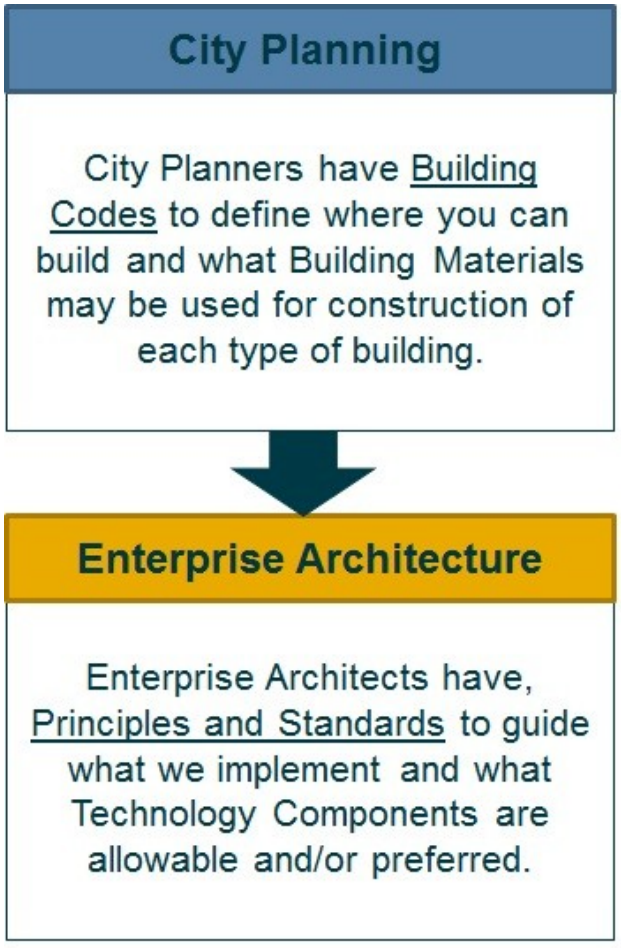
Enterprise Architecture

Enterprise Architects have, Principles and Standards to guide what we implement and what Technology Components are allowable and/or preferred.

Purpose:
To provide documented architectural information that aids managers, project leaders, analysts, and developers in making correct IT choices.

A classic: City Planning Analogy

A classic: City Planning Analogy



Purpose:
To provide documented architectural information that aids managers, project leaders, analysts, and developers in making correct IT choices.

The catch

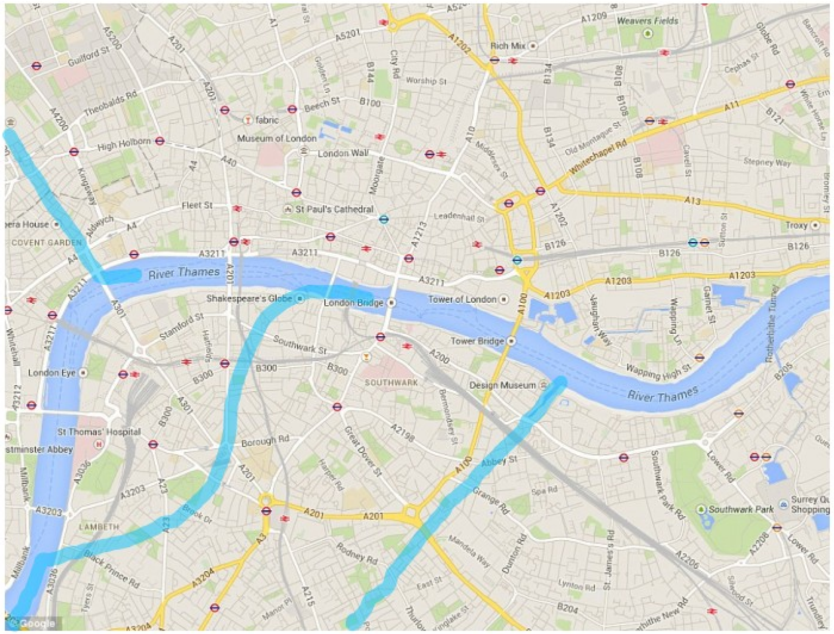
Take a urban layout of a major city.

Look a the river going through it.

Imagine that every 3 weeks, a new affluent, confluent is added or removed.

How do you plan?

How do you operate your city?



An Adaptive System

A Solution Architecture realizes a vision: The life-long purpose and function of an EVOLUTIONARY Architecture is deliver value to its "users" overtime, in an ever-evolving given context of use.

An Architecture solves for the present defined problem: The resulting solution is a set of interacting or interdependent entities forming an integrated whole together respond to environmental changes,

or changes in the interacting parts of the design.

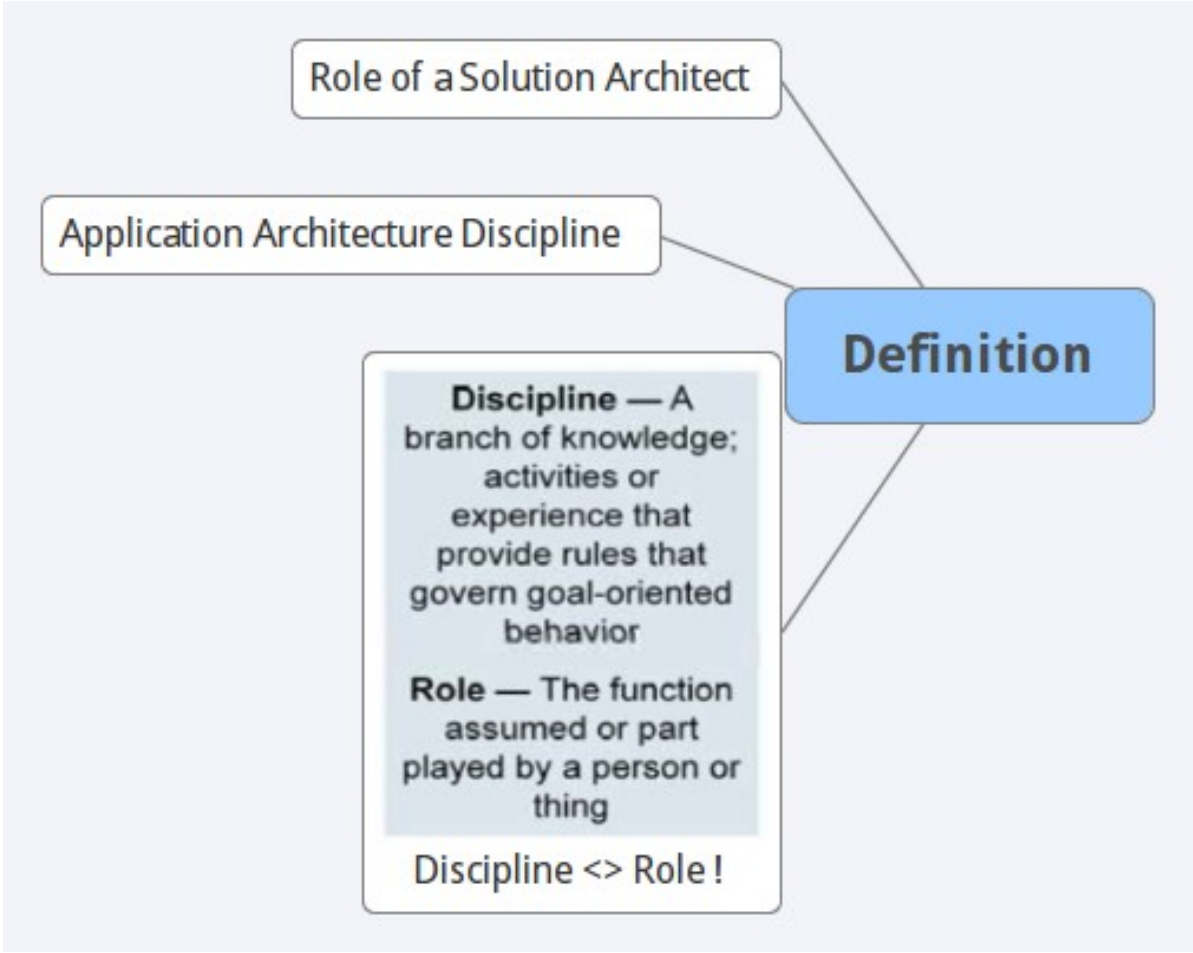
An Architecture must permit extension, refactoring of its constituents to from the existing Solution in an economically viable way, without throwing away the existing system(s).

Diversity of IT Assets

Every application/data solution built, every technology acquired, is here to stay...

...for a while: IT/Systems Assets, IT/Data Assets, Technology Assets

Definition

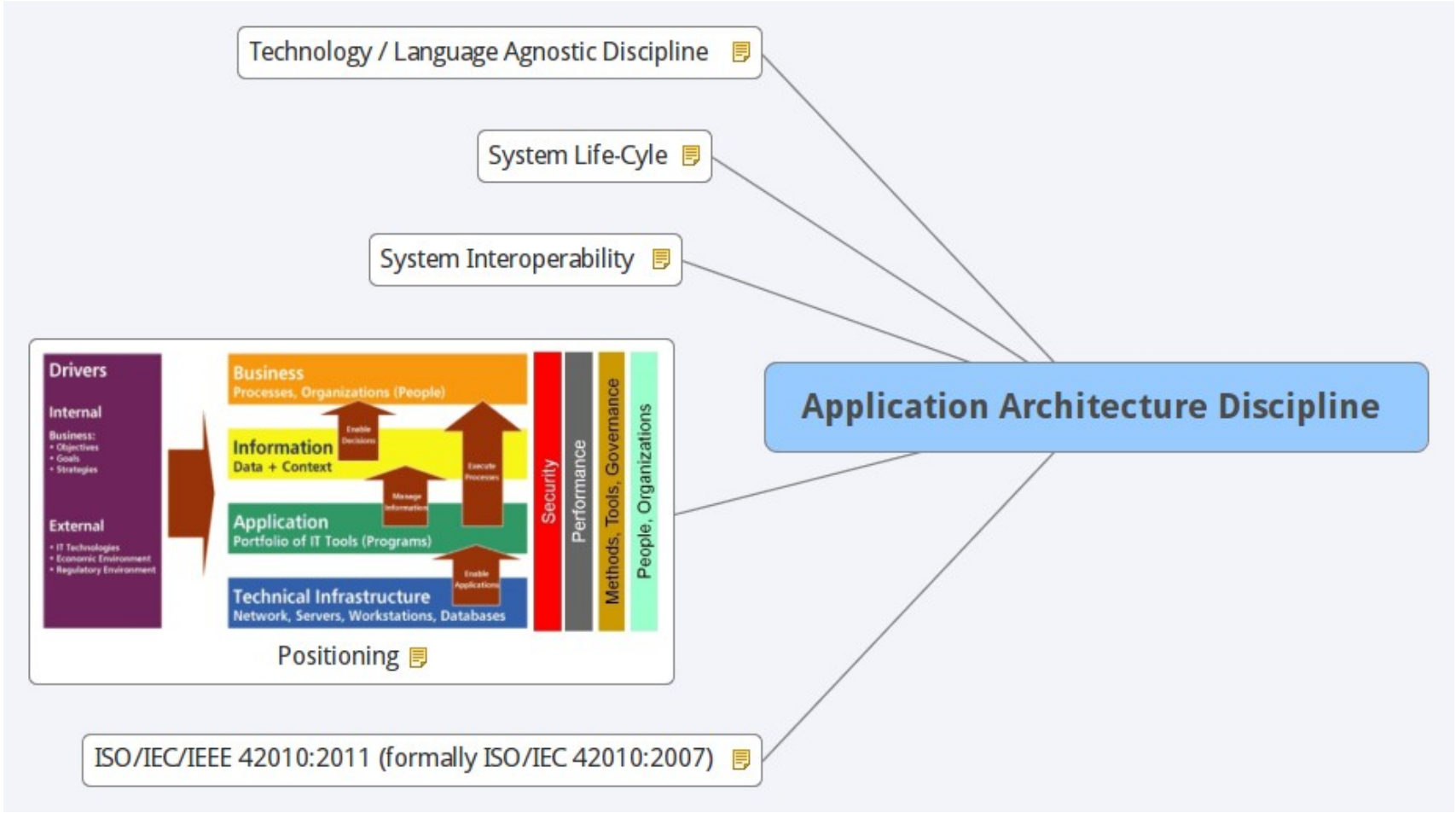


Discipline <> Role !

Discipline — A branch of knowledge; activities or experience that provide rules that govern goal-oriented behavior

Role — The function assumed or part played by a person or thing

Application Architecture Discipline



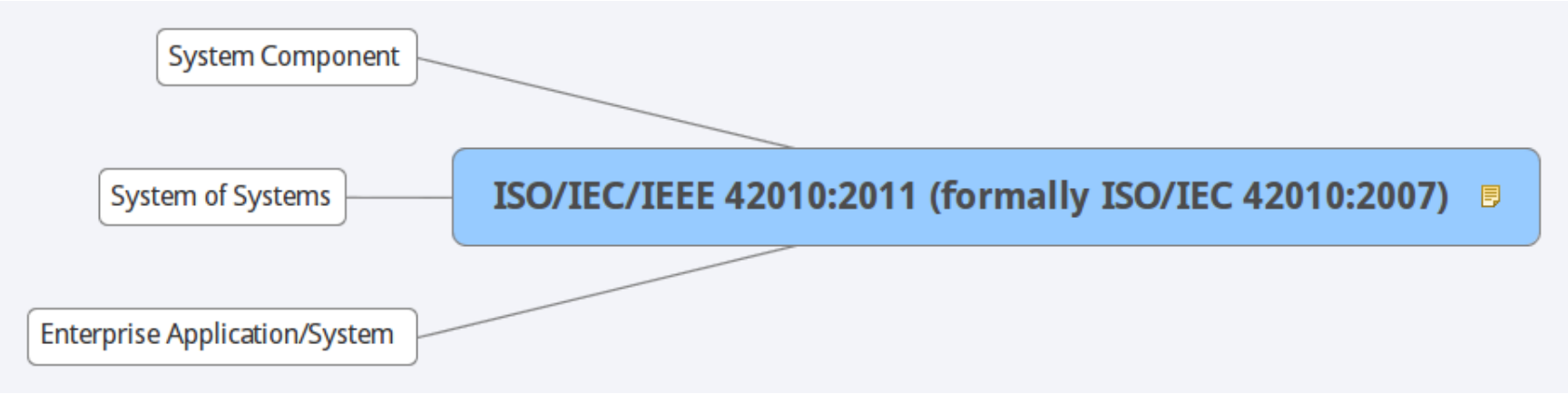
ISO/IEC/IEEE 42010:2011 (formally ISO/IEC 42010:2007)

The Architecture of a SYSTEM is the fundamental organization of that system embodied in its COMPONENTS,

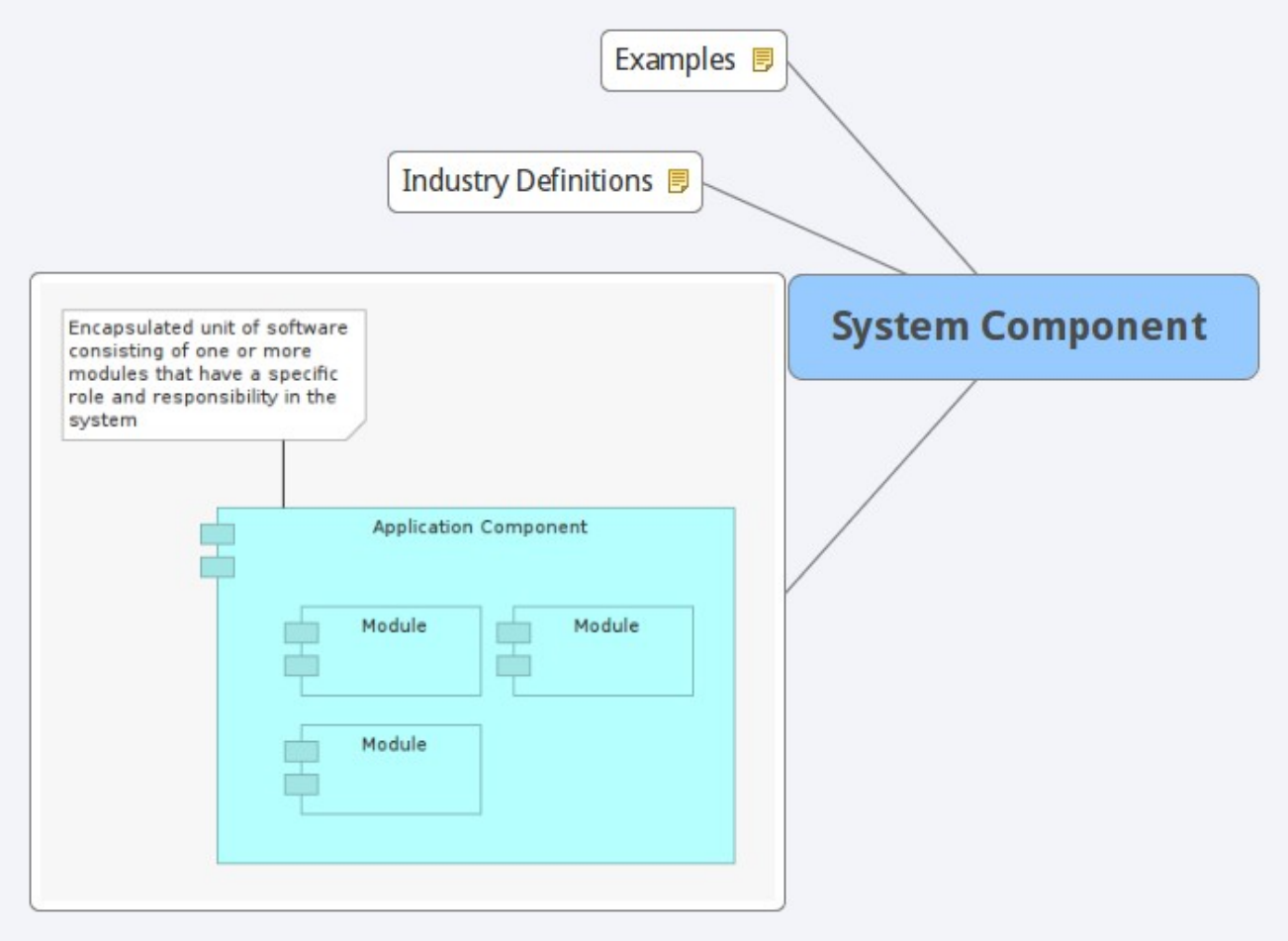
their relationships to each other, and to the ENVIRONMENT,

and the PRINCIPLES guiding its DESIGN and EVOLUTION.

...or simply put: STRUCTURE with a VISION.



System Component



Industry Definitions

"A component is a nontrivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces." (Philippe Krutchen, Rational Software)

"A runtime software component is a dynamically bindable package of one or more programs managed as a unit and accessed through documented interfaces that can be discovered at runtime." (Gartner Group)

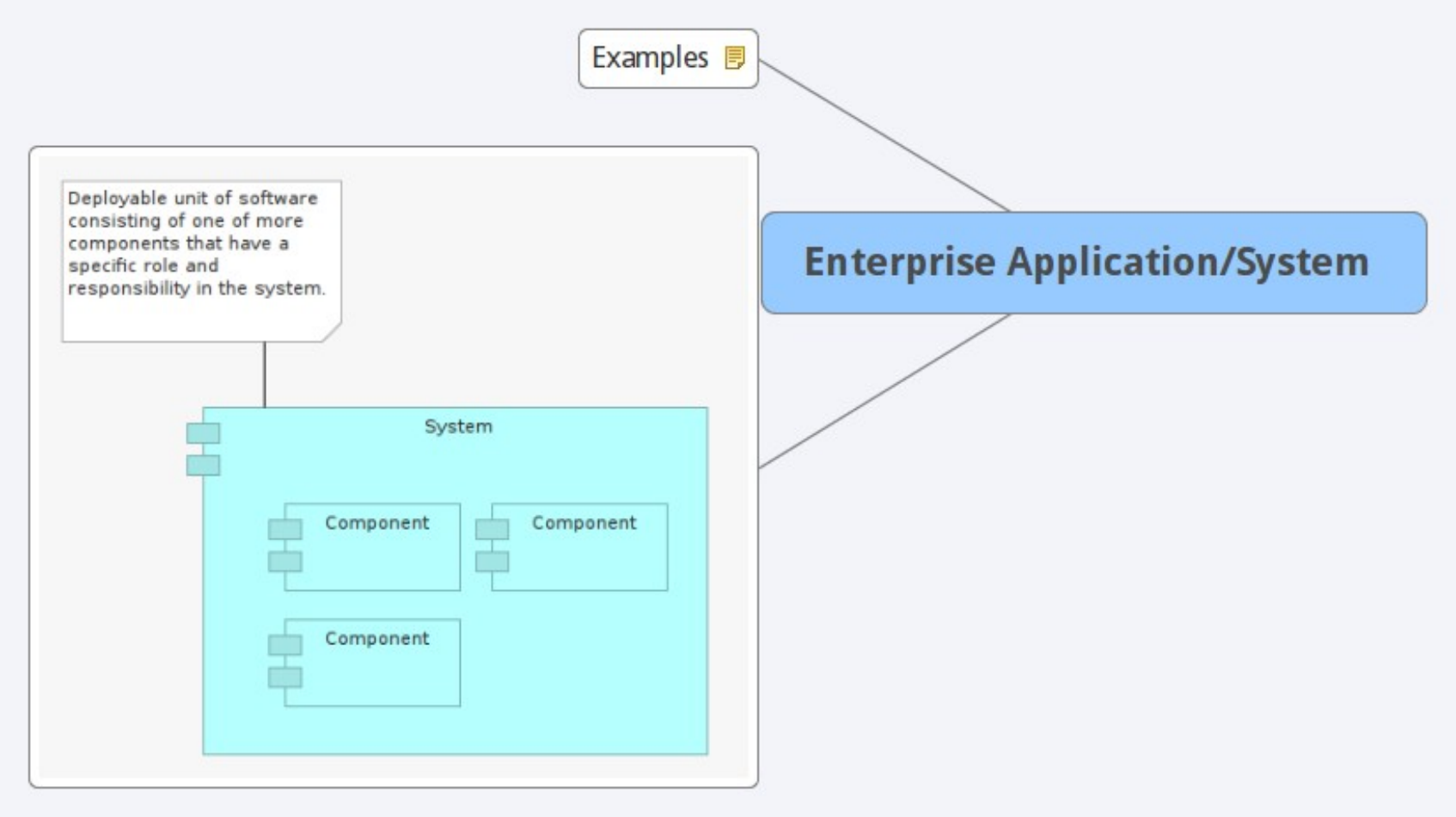
"A component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces...typically represents the physical packaging of otherwise logical elements, such as classes, interfaces, and collaborations." (Grady Booch, Jim Rumbaugh, Ivar Jacobson, The UML User Guide, p. 343)

Examples

Functional Components: ex. A pricing engine

Technical Components: ex. logging/auditing

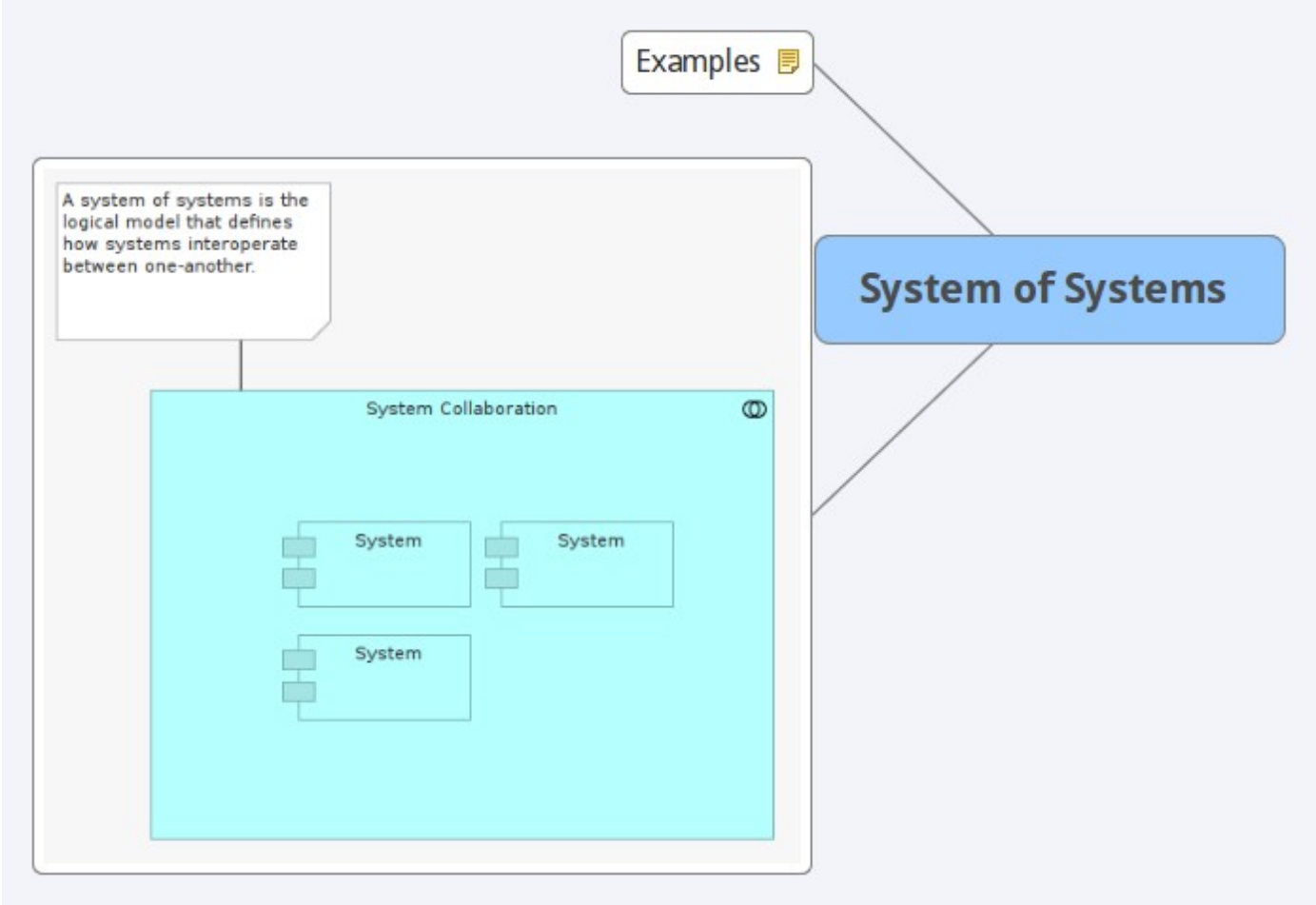
Enterprise Application/System



Examples

Customer Relationship Management System
Customer Service Support System
Policy Administration System
Claims Management System

System of Systems



Examples

Enterprise Resource Planning

- Sales Orders, Contracts, Quality management, Pricing (...)

Supply Chain Management

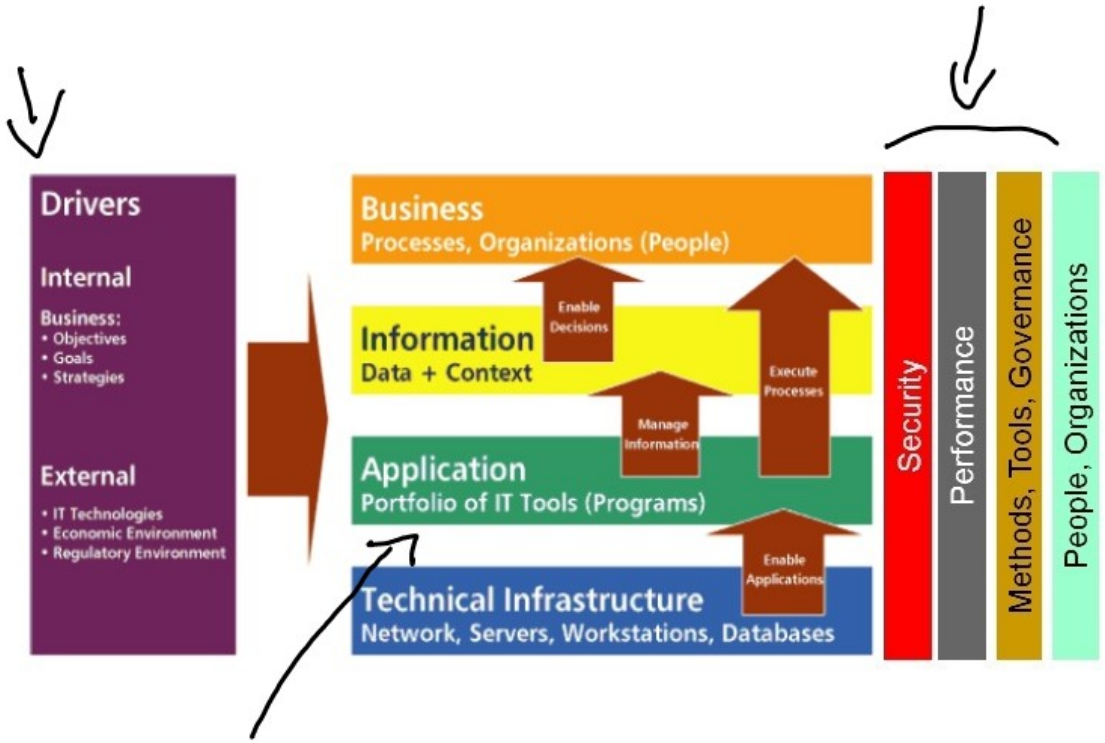
- Product development, Material sourcing, Production and logistics (...)

Financial / Accounting

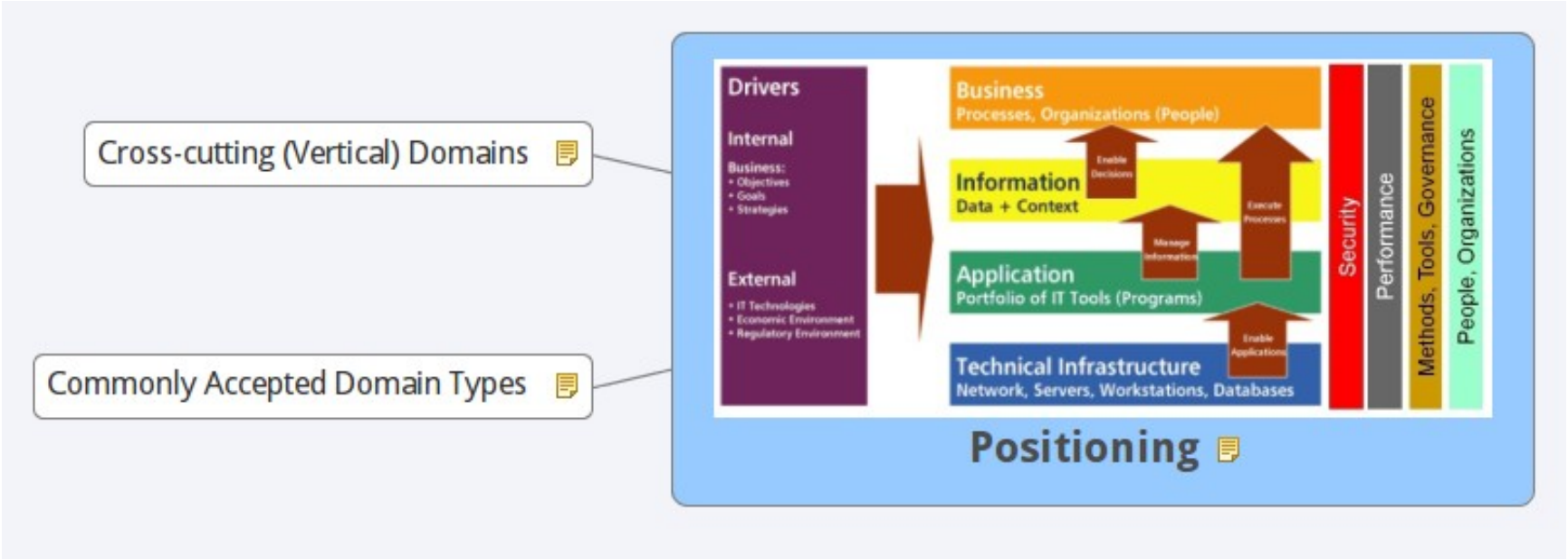
- Accounts Receivable, Accounts Payable, Finance Data warehouse, Finance Reporting, General Ledger (...)

... including surrounding information systems that coordinate these activities.

Positioning



Positioning



Positioning

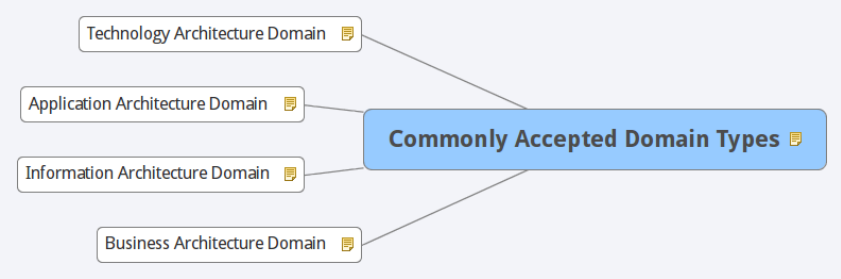
The classification and specialization of skills, techniques, tools, terminologies in Architectural Domains aim to achieve: (a.) A formal description of a system, or a detailed plan of the system at component level to guide its implementation, and (b.) To "structure the components of a System, their inter-relationships, and the FORCES, principles and guidelines governing their design and evolution over time.

Commonly Accepted Domain Types

All Architectural Domains describe the Operating Model (OM) of the Enterprise As-a-System using 4 Domains of skills/expertise.

These Architectural Domains inform the Operational Transformation (OT) required to support the Strategy moving the Enterprise to a Target State Architecture.

All Architectural Domains propose tangible deliverables, falling into two broad categories: (a.) Asset Inventories, (b.) Solution Designs.



Business Architecture Domain

The Business Architecture Domain describes the OM and OT of an Enterprise under the lenses of:

(a.) its Business Capabilities,

(b.) its Value Proposition Streams,

(c.) the classification of its Information Assets,

(d.) its Organizational design.

Information Architecture Domain

The Information Architecture Domain describes the OM and OT of an Enterprise under the lenses of:

(a.) the Structure of its logical and physical Data assets,

(b.) the definition of its Data-interchange Standards and Contracts for Systems to productively inter-operate,

(c.) the mechanisms guaranteing the Integrity & Reliability of its Enterprise Information Assets.

Application Architecture Domain

The Application Architecture Domain describes the OM and OT of an Enterprise under the lenses of:

(a.) the structure of its Systems & constituting Components, answering both Functional and Non-functional requirements,

(b.) the responsibility of such Systems & constituting Components, supporting the execution of Business processes,

(c.) the interactions between such Systems & constituting Components, the amount and nature of then relationships between each,

(d.) the life-cycle and deployment of such Systems & constituting Components, on target Devices and Technology host platforms.

Technology Architecture Domain

The Technology Architecture Domain describes the OM and OT of an Enterprise under the lenses of:

(a.) the infrastructure capabilities required to support the deployment of business, data, and application solutions, from a Hardware and Software standpoint,

(b.) the introduction, governance and management of the life-cycle for net new & existing Technology assets,

(c.) the optimized management of infrastructure costs (i.e. middleware, networks, communications, storage & processing power), from the standpoint of "utilities".

Cross-cutting (Vertical) Domains

Transversal to Architectural Domains that are Internal Drivers (Goals, Objectives), and External Drivers (Market Forces).

Transversal to Architectural Domains are Perspectives answering Non-functional requirements, Technology Trends (i.e. Hype-Cycle).

System Interoperability

Architecture Components interoperate with other components by exchanging information/data.

System Life-Cyle

Architecture Components have one or many of the following characteristics:

- REUSABLE units, library, runtime library, vendor platform
- AUTONOMOUS, deployable, centralized
- BLACK-BOX

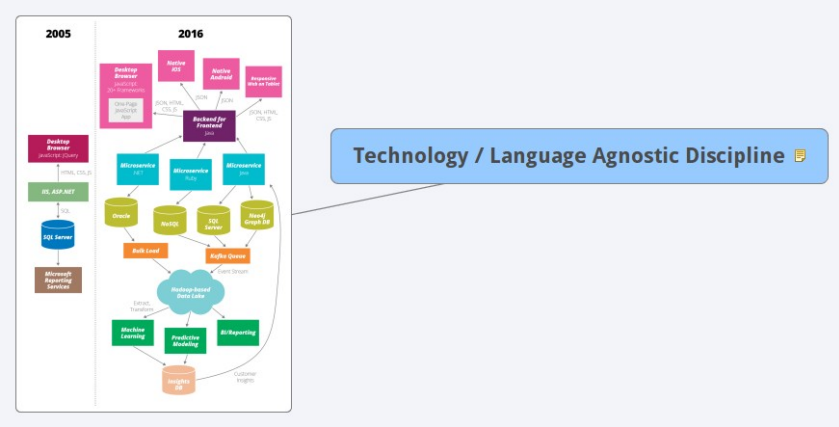
Language Agnostic Discipline

Architecture is language agnostic.

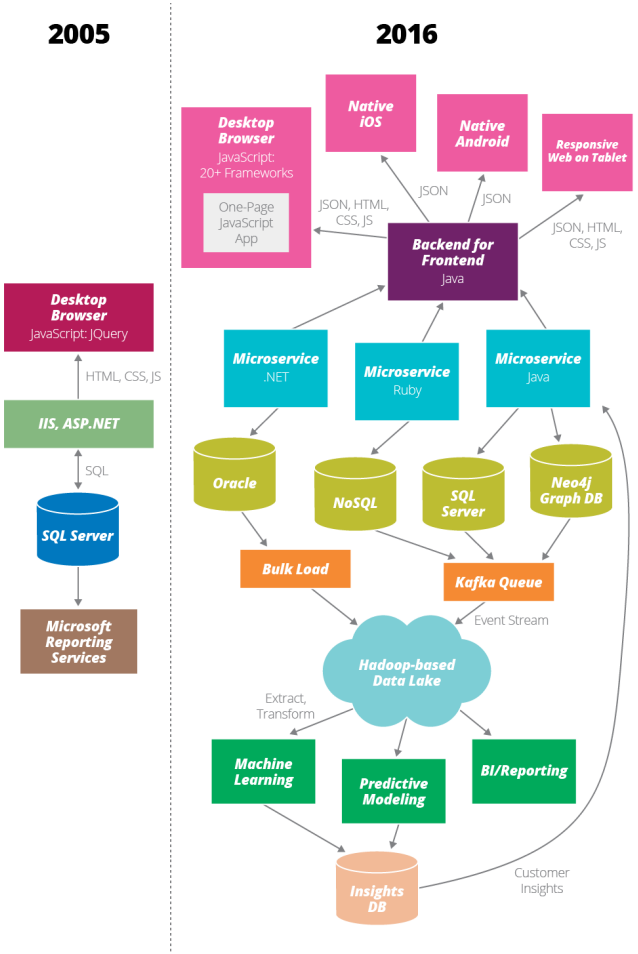
The modeling bricks of Architecture are System Components, Systems, System of Systems.

Making sense of Architecture requires putting the above into CONTEXT: (1.) a structure, (2.) a flow ...

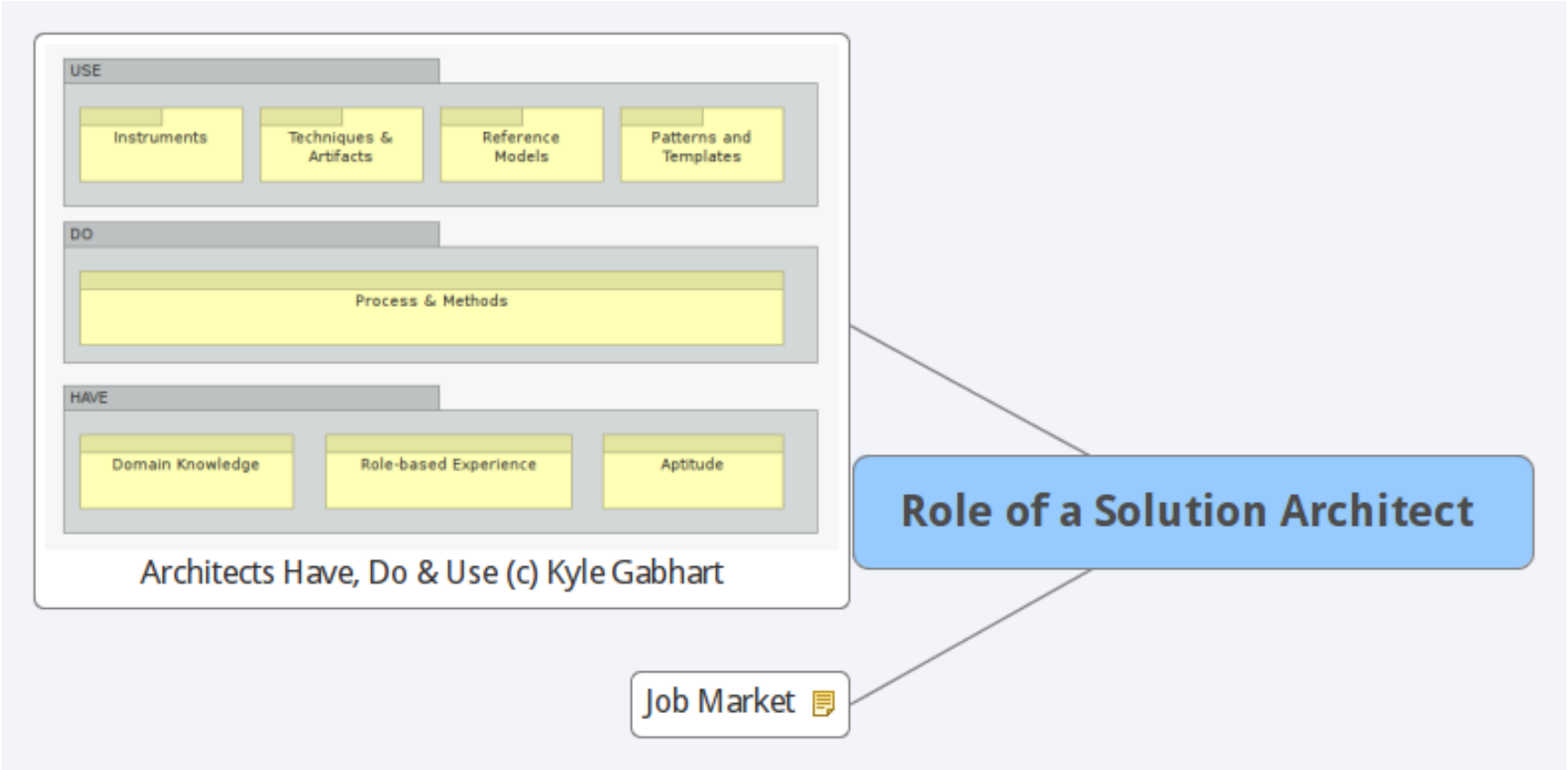
...aiming to SOLVE for a purpose/vision, or SOLVE for a problem.



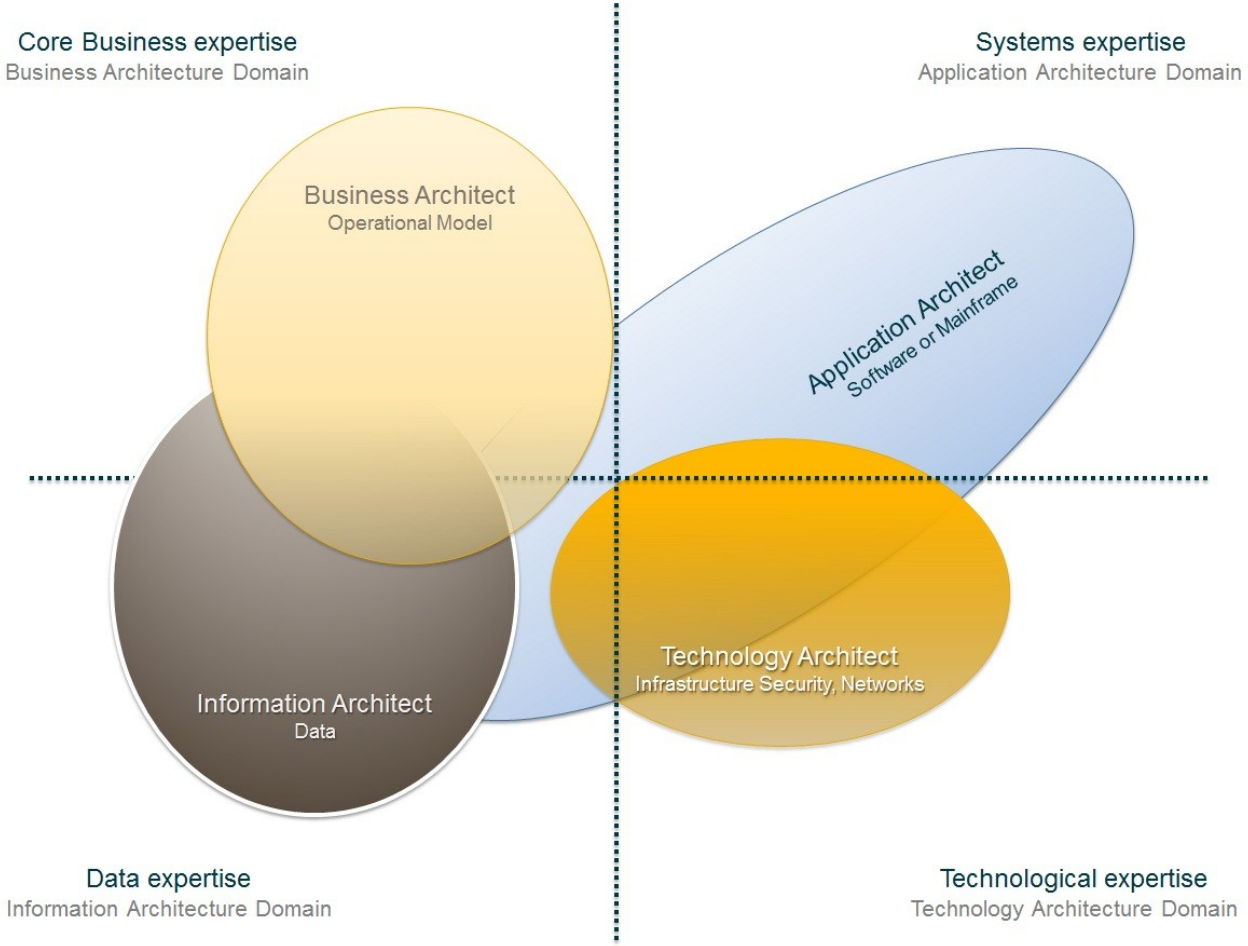
Greatly influenced by Technology



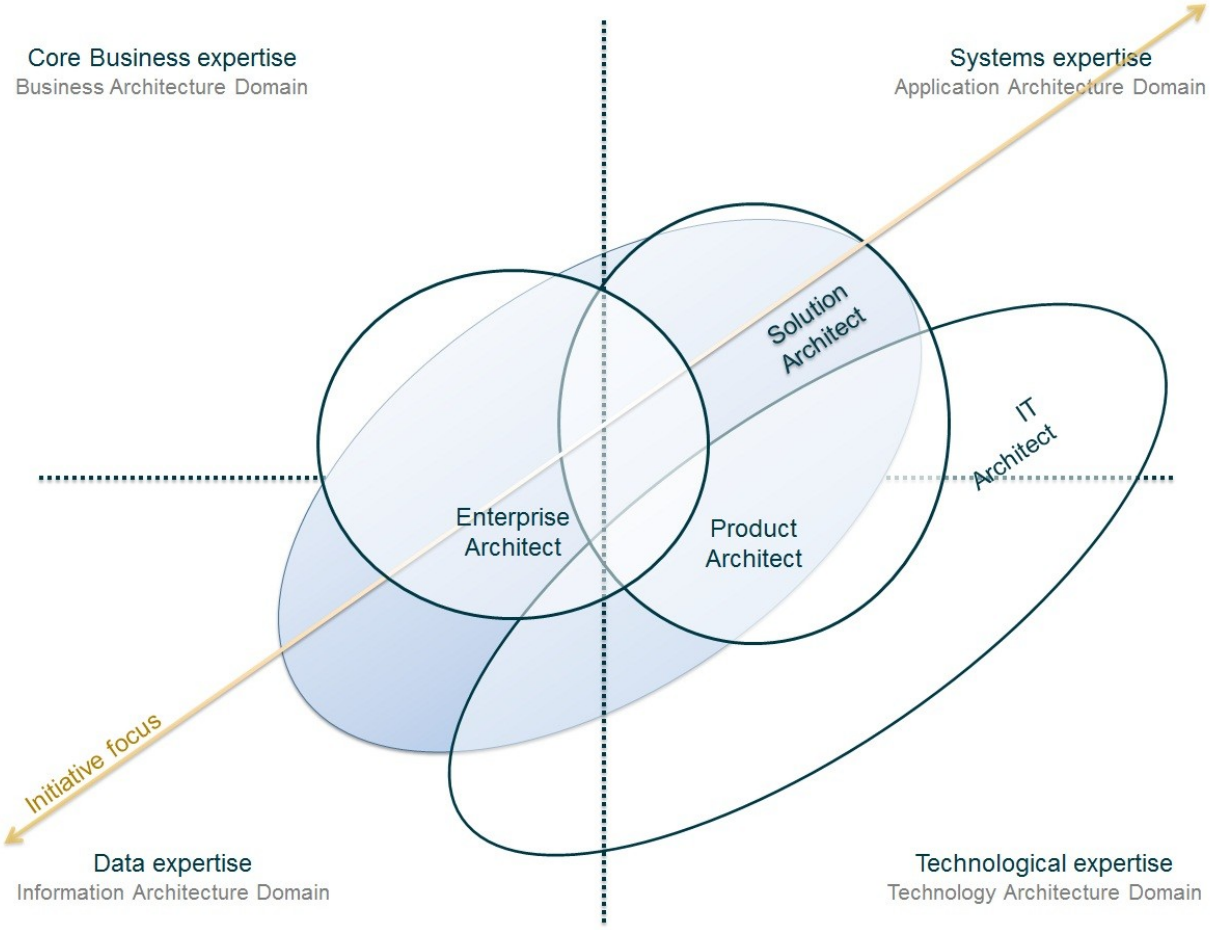
Role of a Solution Architect



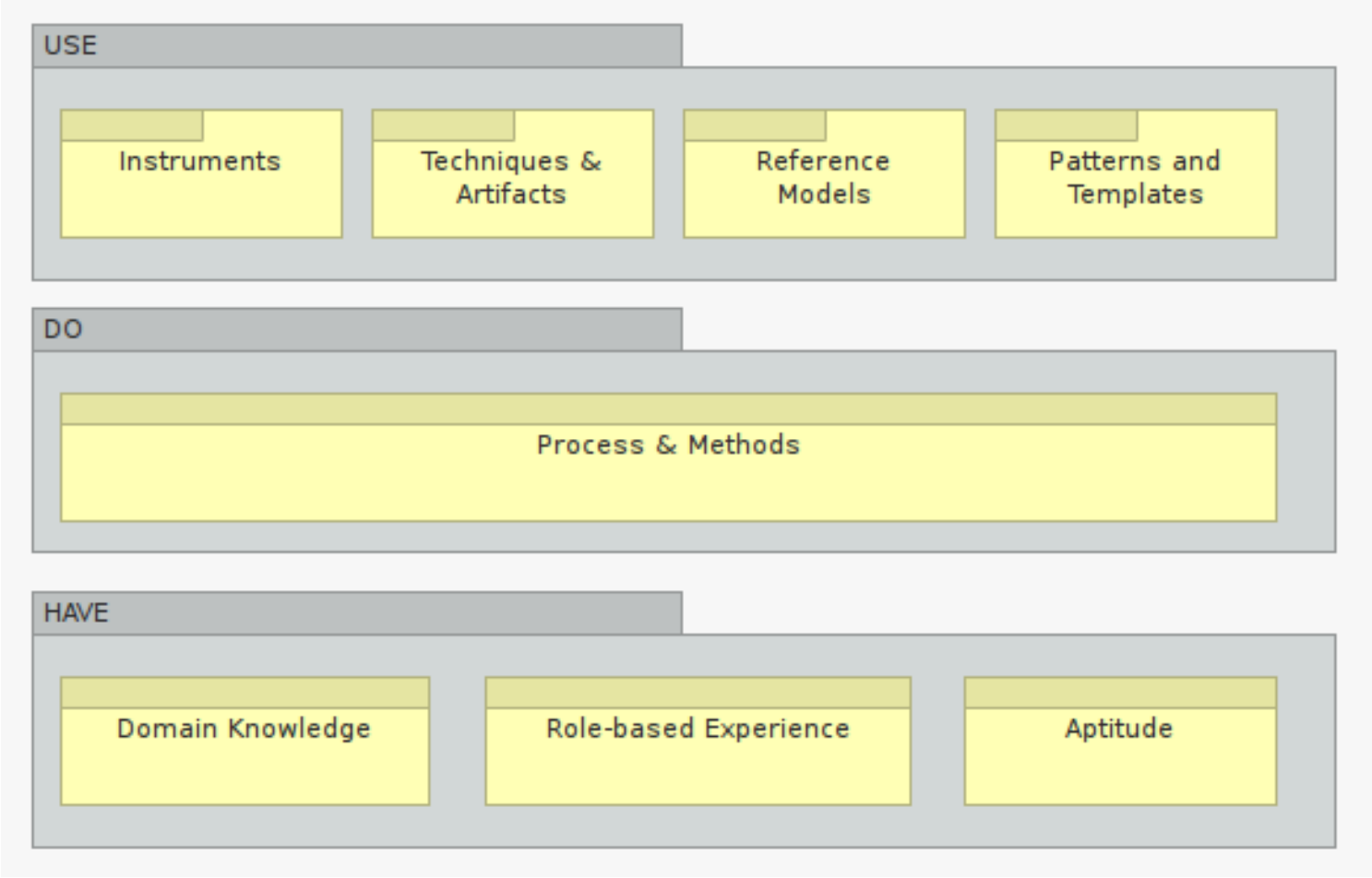
Domain Role



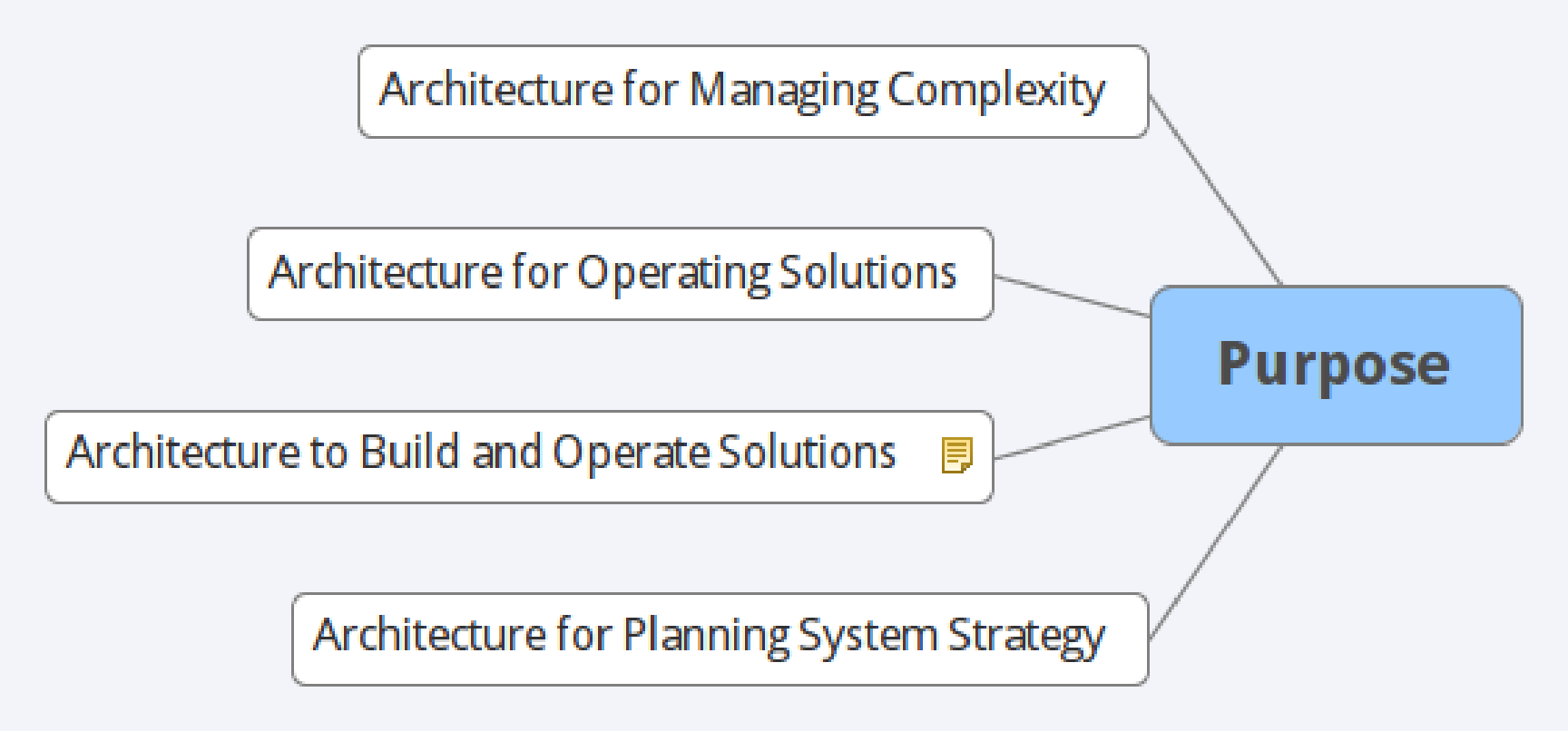
Situational Role



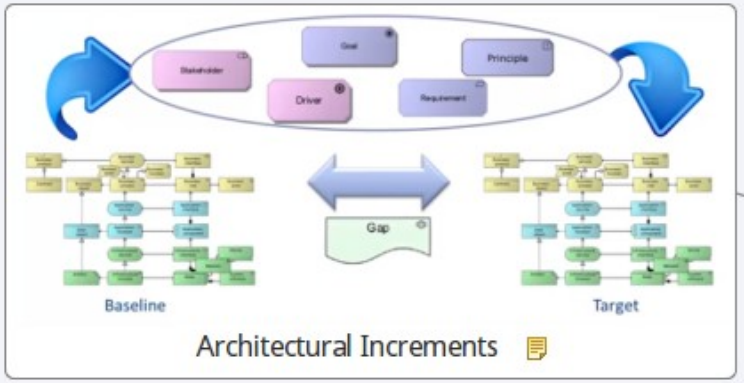
Architects Have, Do & Use (c) Kyle Gabhart



Purpose of Application Architecture



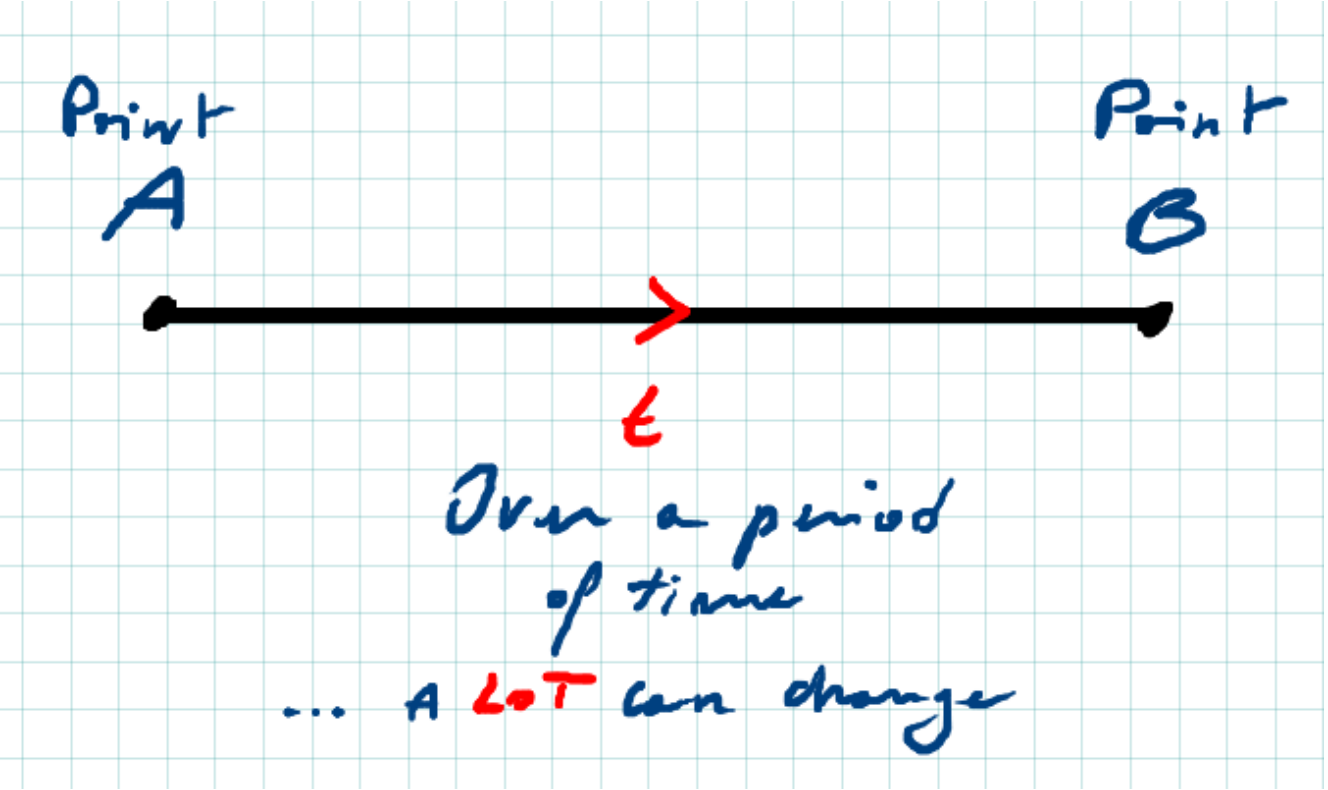
Architecture for Planning System Strategy



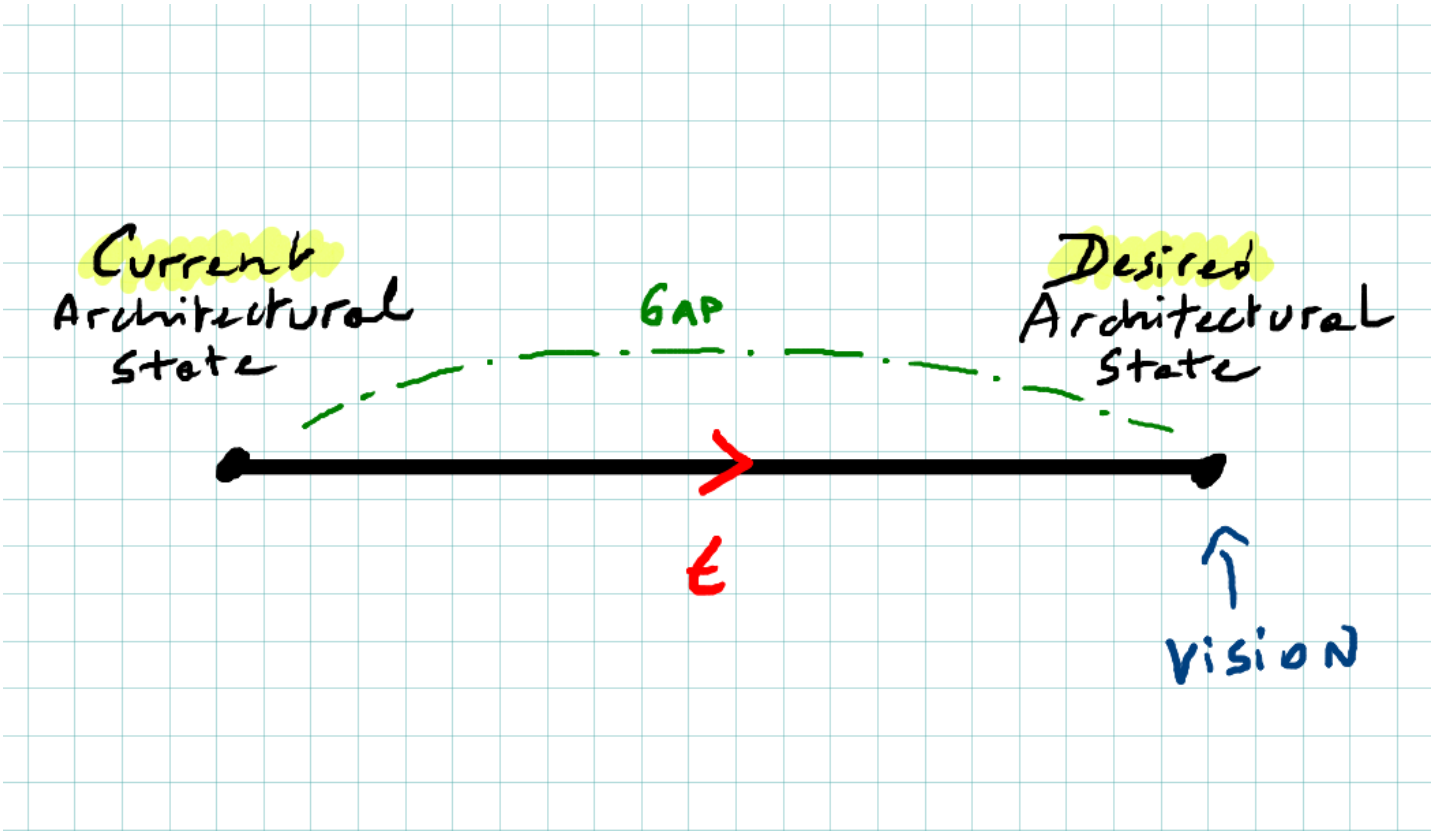
Architecture for Planning System Strategy

About "Strategy"

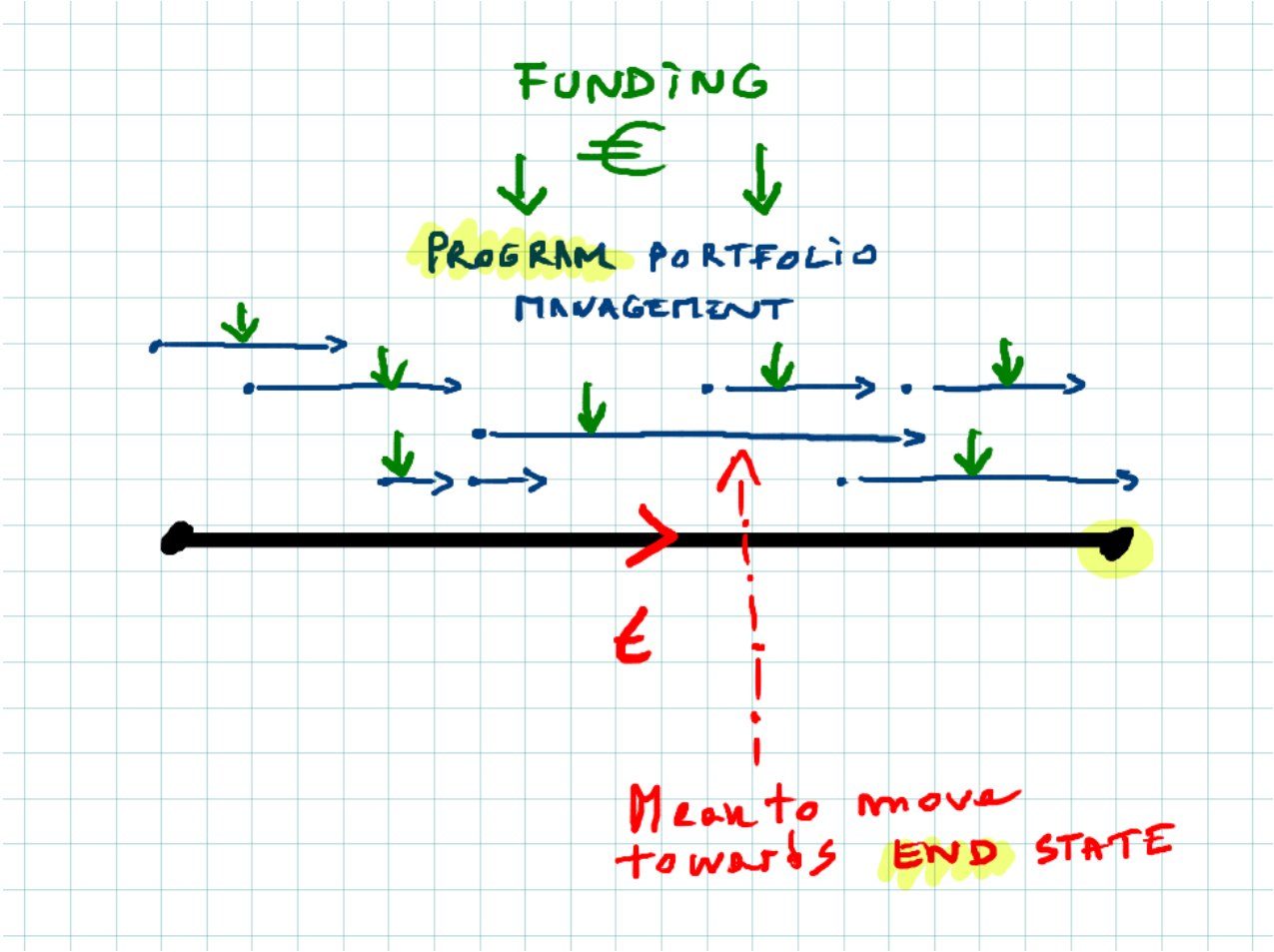
Strategy to go from A to B



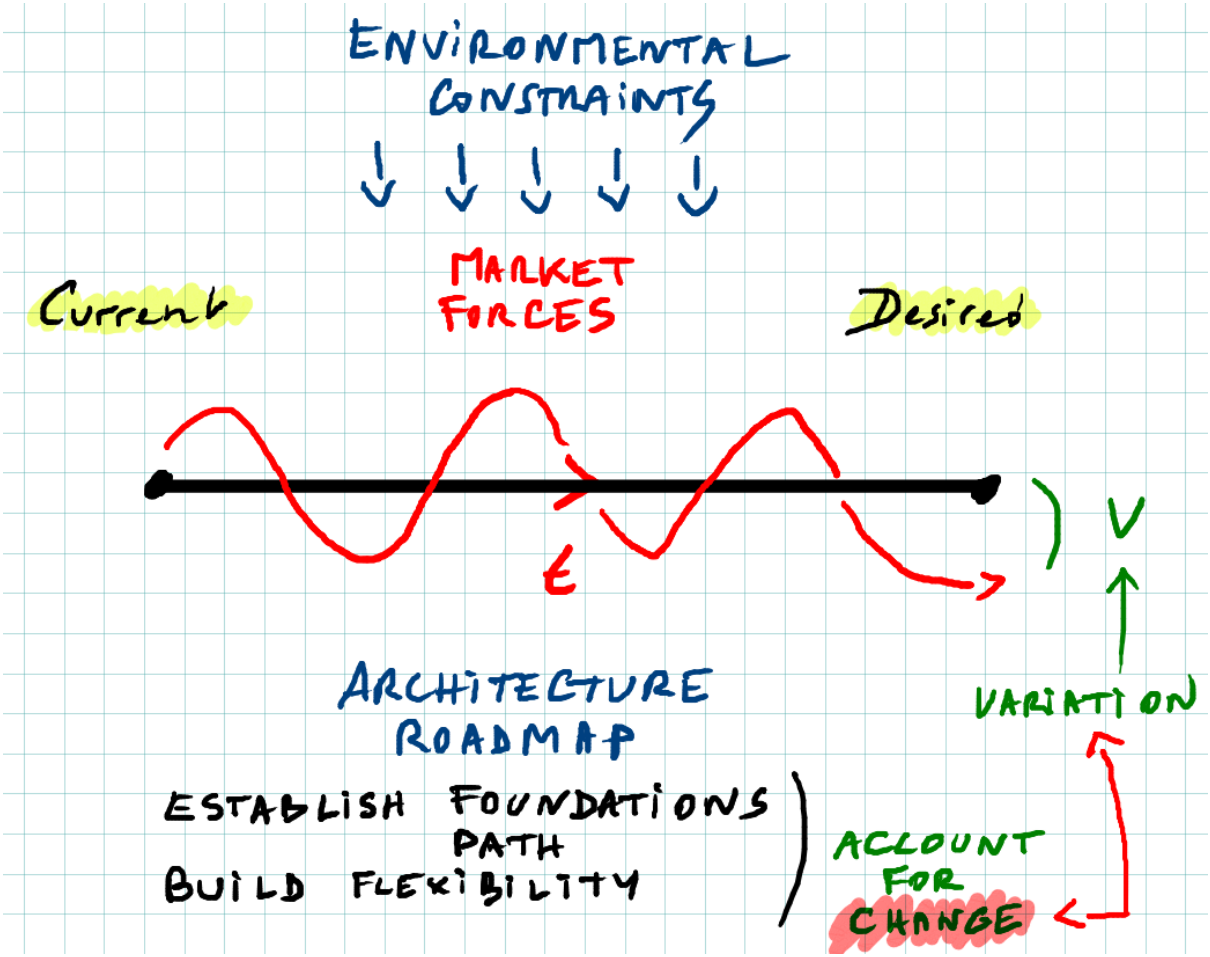
Strategy: To realize a Vision of the Enterprise



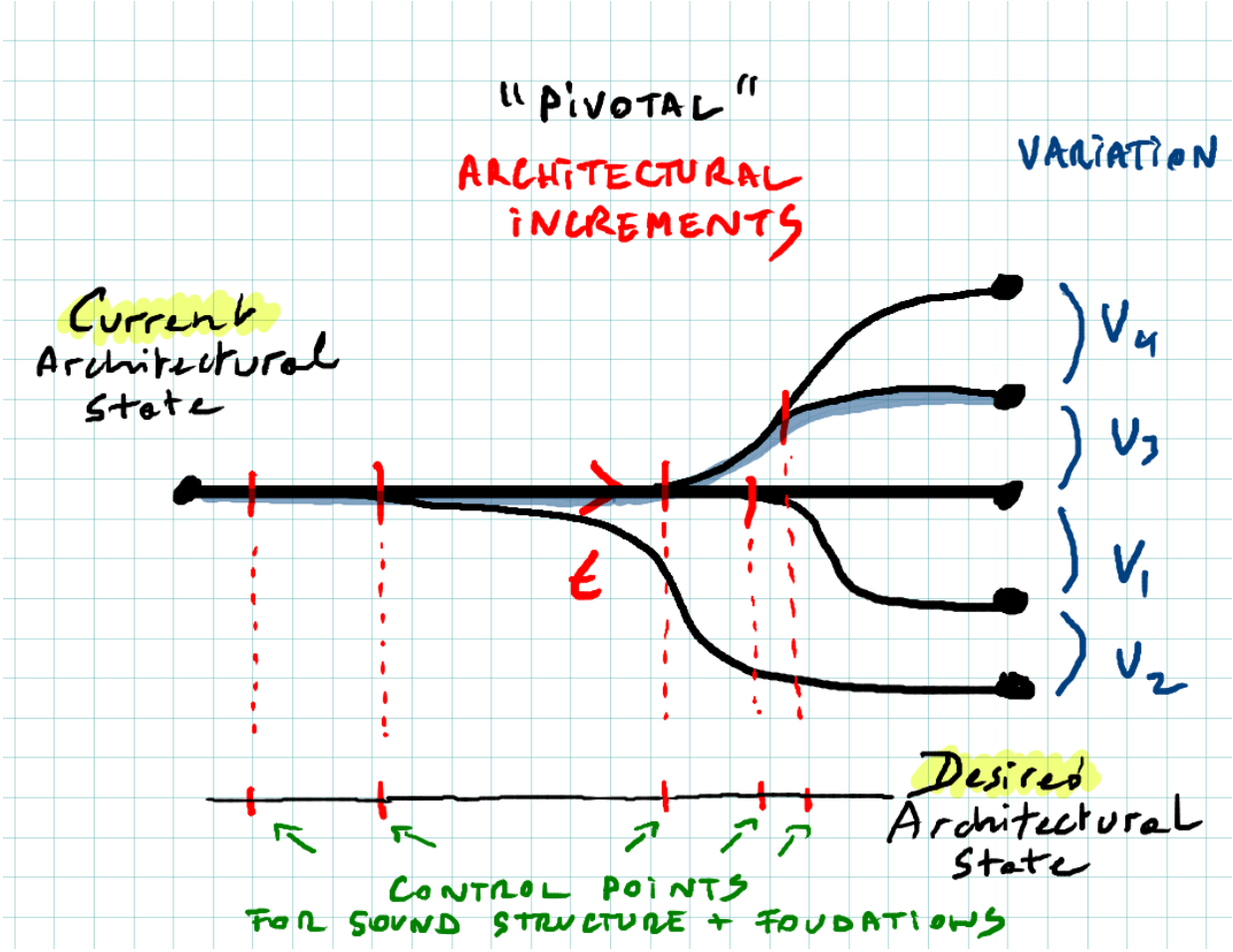
Means of moving a Strategy forward



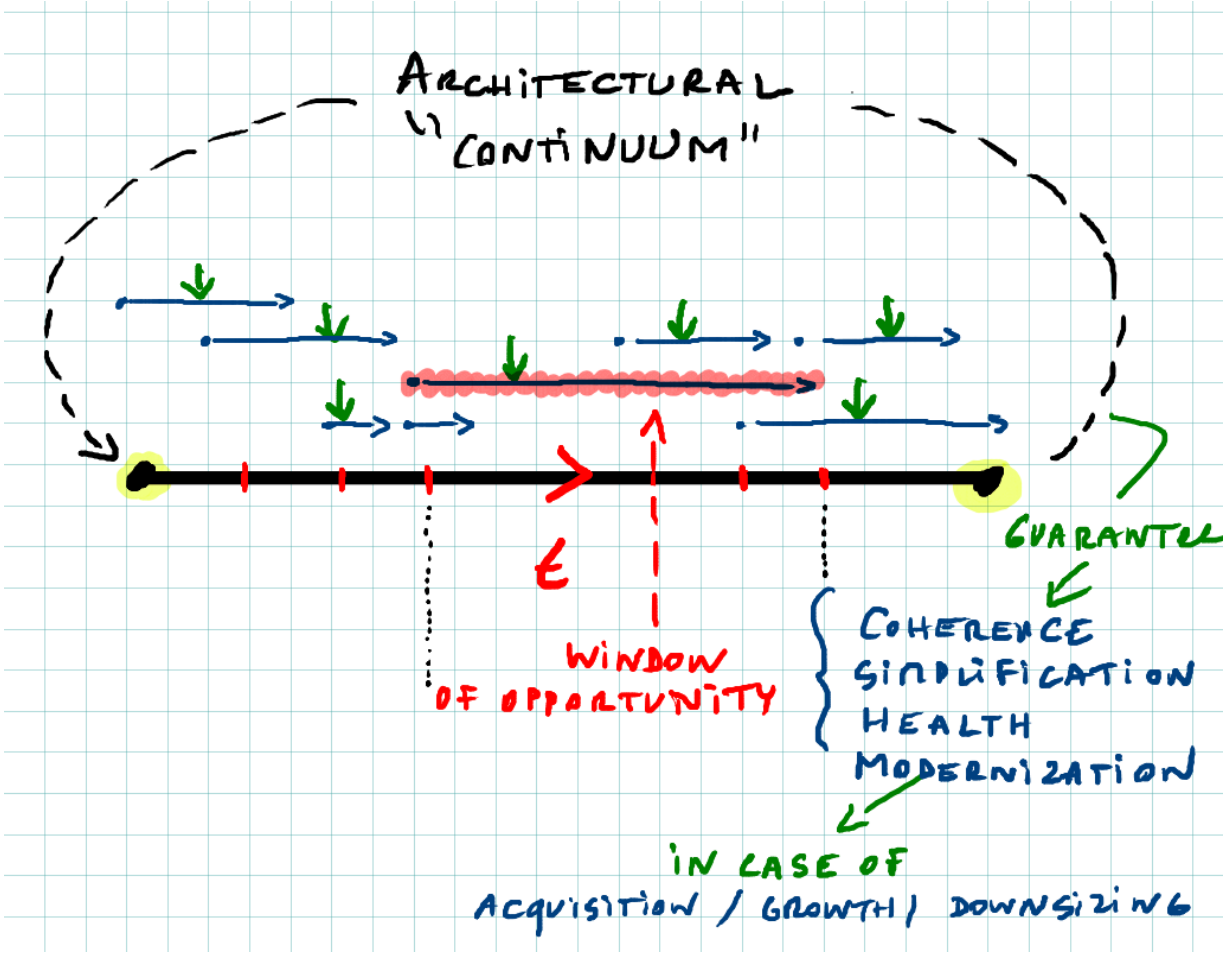
"Managing Complexity and Change" of Strategy



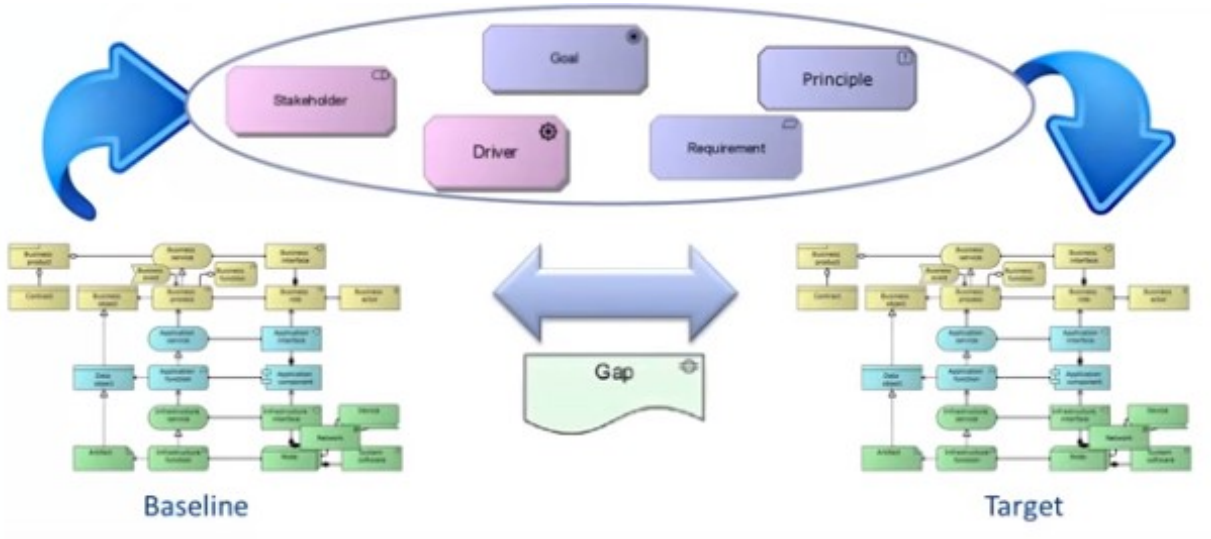
Architectural Governance to assess impact on Strategy variations



Continuum to protect an Enterprise from itself



Architectural Increments



Architectural Increments

The next natural step after strategy

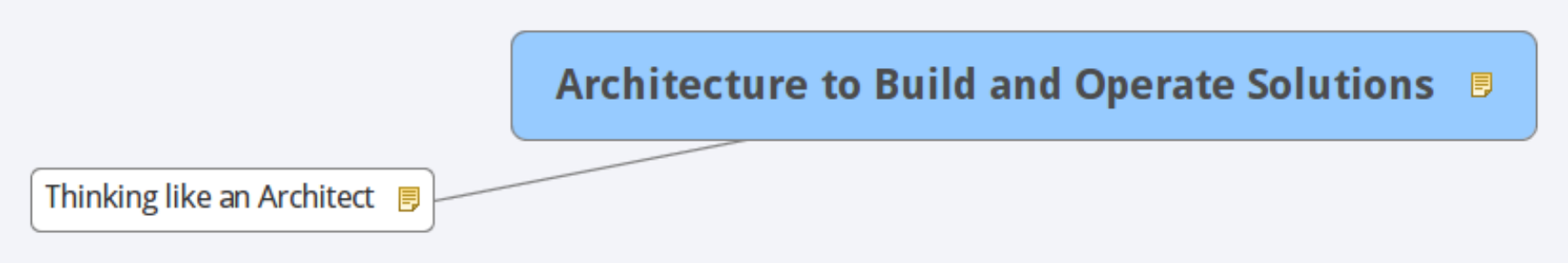
Provide alternative to decommission "aging" systems

Architecture to Build and Operate Solutions

Architecture tries to AVOID having to delve into or reverse engineer existing applications/systems to understand what they do, and how they do it.

Architecture groups modules into components, components into systems, systems into services.

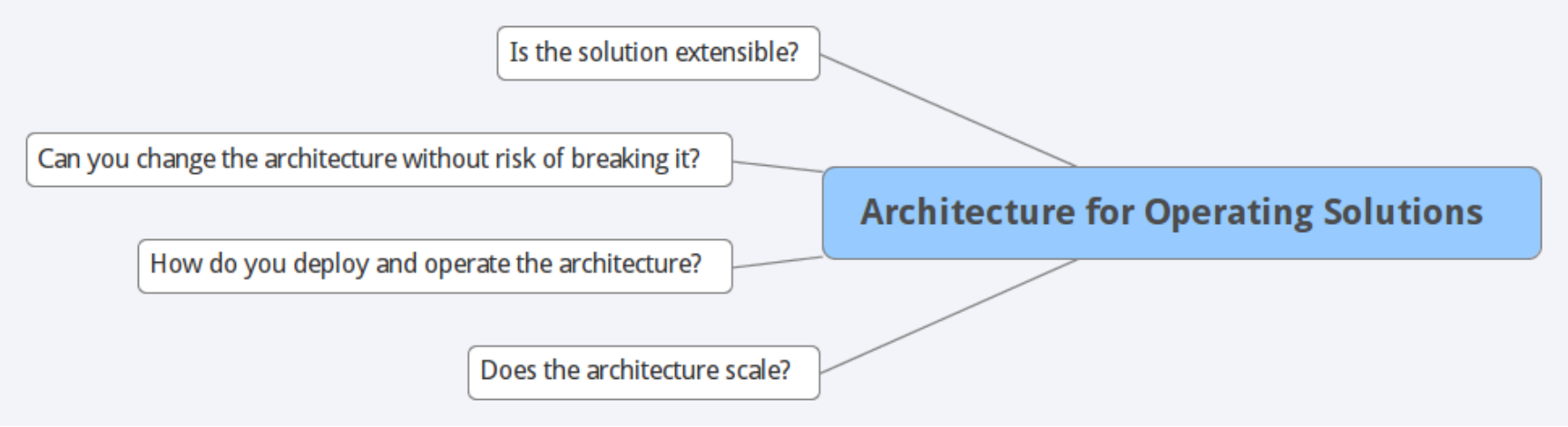
Architecture provides deep insight on the structure and behavior of components, systems and how they interoperate between one another.



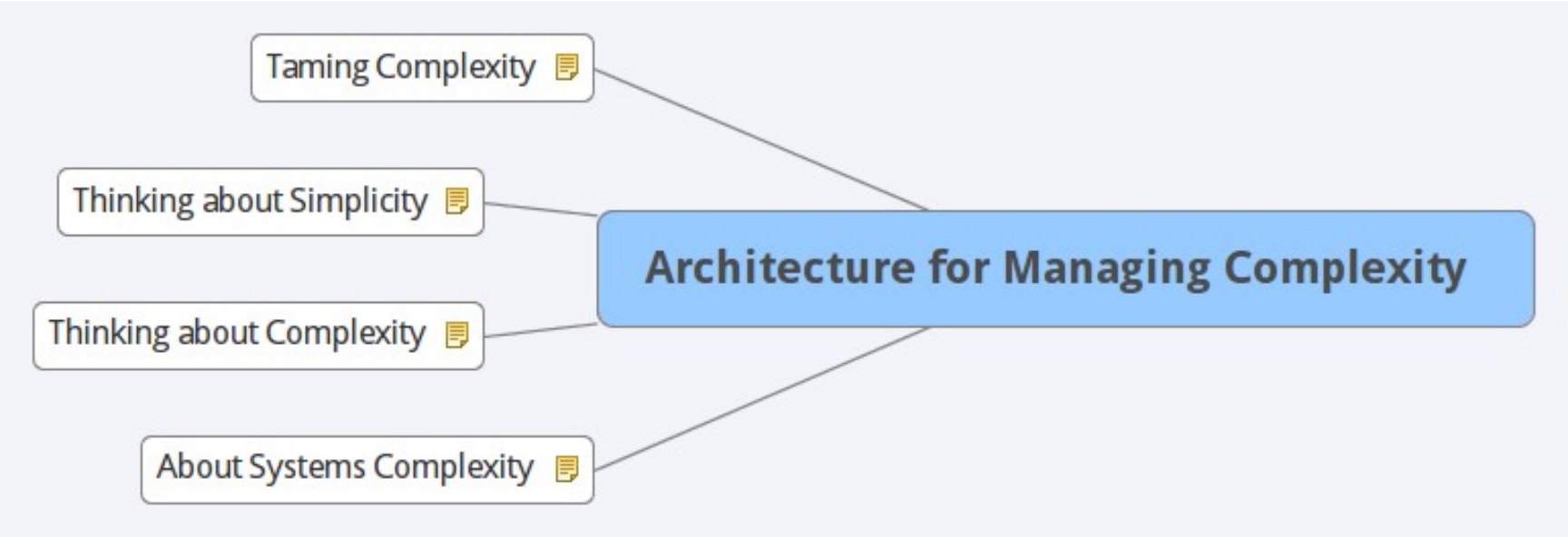
Thinking like an Architect

Incremental specifications, with the expectation of spiraling back for changes, is the best approach to architecting. A complex problem cannot be solved all at once. It requires an incremental approach. As the architect knows more about a solution, that knowledge is folded back into the specification. How an architect determines when to drill down into a part to more fully specify it, or to address all parts at a coarse-grained level, is an individual decision. It doesn't matter how an architect approaches the specification of an architecture, as long as the system functionality and semantic behavior are in the end well-defined: traceable decision, complete, fit for purpose.

Architecture for Operating Solutions



Architecture for Managing Complexity



About Systems Complexity

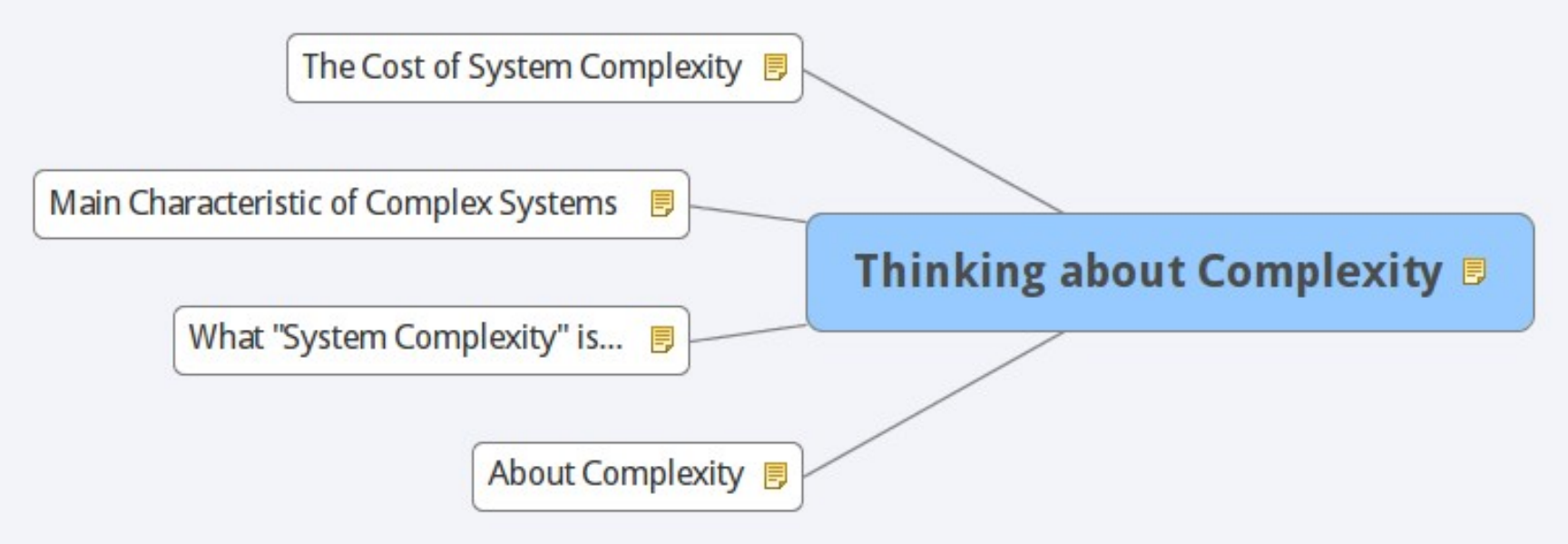
Architecture hunts Complexity both in the domains of Systems and Process design.

Architecture aims to tame Systems Complexity.

Badly designed Architectures is a source of more complexity.

Thinking about Complexity

"Complexity is the enemy of computer science, and it behooves us, as designers, to minimize it." - Charles Thacker, CACM, July 2010.



About Complexity

Complexity arises in situations where “an increasing number of independent "things" begin interacting in interdependent and unpredictable ways.

In IT, "things" are Software features, Hardware features, Users, Processes, other Systems...

What "System Complexity" is...

Intrinsic Systems Complexity is induced by: (1.) Scale: Many “things”, (2.) Diversity: Many different kinds of “things”, (3.) Interconnectivity: Many “things” connected to many things in different ways.

Extrinsic Systems Complexity is induced by: (1.) the level of integration its contracts with other "things" (autonomy), (2.) its sensibility to external or internal triggers (behavior rules and timing for state change), (3.) its sensibility to external constraints (reliability and dependability).

Main Characteristic of Complex Systems

Failures, characterized by: (1.) Frequency of failures, (2.) Difficulty of root-cause analysis, (3.) Difficulty of resolution.

Entropy, characterized by: (1.) Technical Debt and Inertia, (2.) Cost of change, negative CBA.

Scale, characterized by: (1.) The System does everything for anybody, (2.) Difficulty to discern boundaries.

Low Predictability of: (1.) behavior, (2.) output, due to: (1.) Behavior is defined by a rich hierarchy of static/dynamic decision rules (a small change may have a large impact), (2.) System internal state is function of past states, (3.) System is a composition of interdependent Systems, or aggregation of smaller, nested Systems, (4.) Wide range of peer-to-peer communications exists between "things", using disjointed communication protocols.

The Cost of System Complexity

Derails Business initiatives from their original intentions.

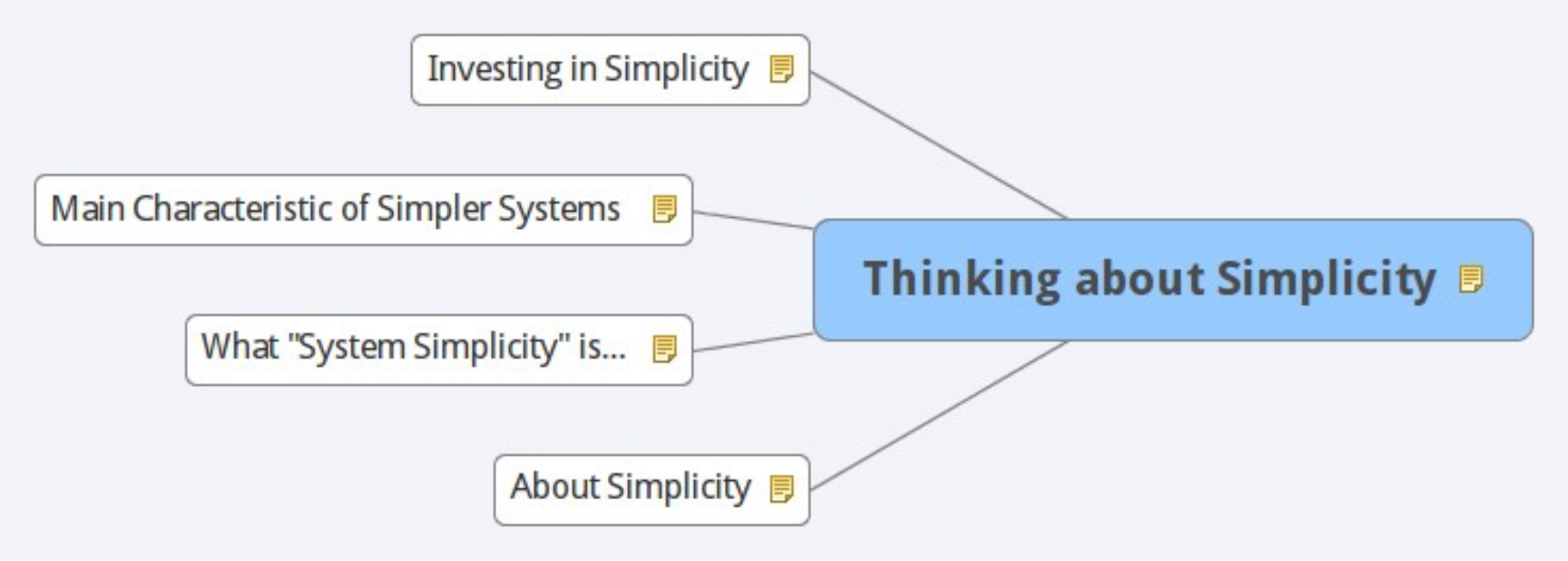
Creates pockets and silos, in turn creating dysfunctional organizations.

Prevents a Firm to quickly adjust to change.

Delivers diminishing returns for high cost of maintenance.

Thinking about Simplicity

"Make things as simple as possible, but no simpler." - Albert Einstein



About Simplicity

Simple is NOT simplistic.

It is designed (balanced), and managed (controlled) complexity.

Increasing complexity is the result of an Evolutionary process keeping demands in check, and focusing on what matters/is significant first and foremost, leaving the unnecessary behind.

What "System Simplicity" is...

System Simplicity is visible in situations where a "limited" number of independent things interact in a "limited" number of interdependent ways.

Intrinsic Systems Simplicity: Limitation of the number of "things".

Extrinsic Systems Simplicity: Limitation of Environmental Constraints.

Simpler Systems address limited scale of related concerns, so their fitness for purpose is optimal.

Main Characteristic of Simpler Systems

Illusion of Simplicity for Stakeholders.

Simpler Systems behave the way you expect it to work, providing increased perceived determinism of behavior and outputs.

Low coupling between "things" due to clear separation of concerns.

Formalized, Centralized, Translated, Monitored, communications.

Investing in Simplicity

A Firm invests in minimizing Complexity.

It does so by meeting the requirements of a Maturity level.

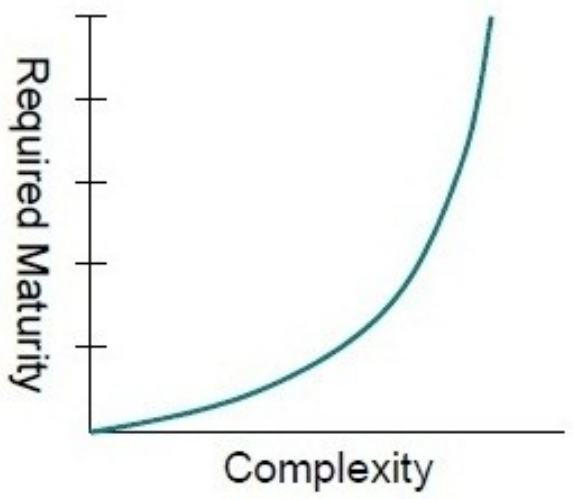
A more complex Enterprise requires a higher Maturity Level.

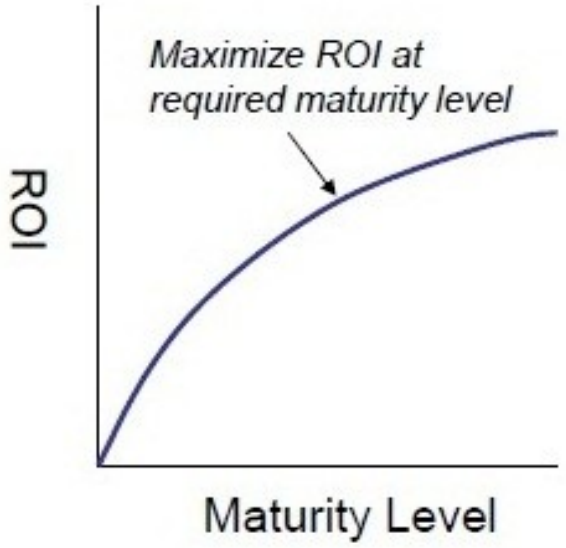
"What" level of Maturity is required for a Firm, "When"?

Then a Firm can answer what is the optimal level of investment within those Maturity requirements?

Anything else than optimal triggers the "Law of diminishing returns".

Law of Diminishing Returns



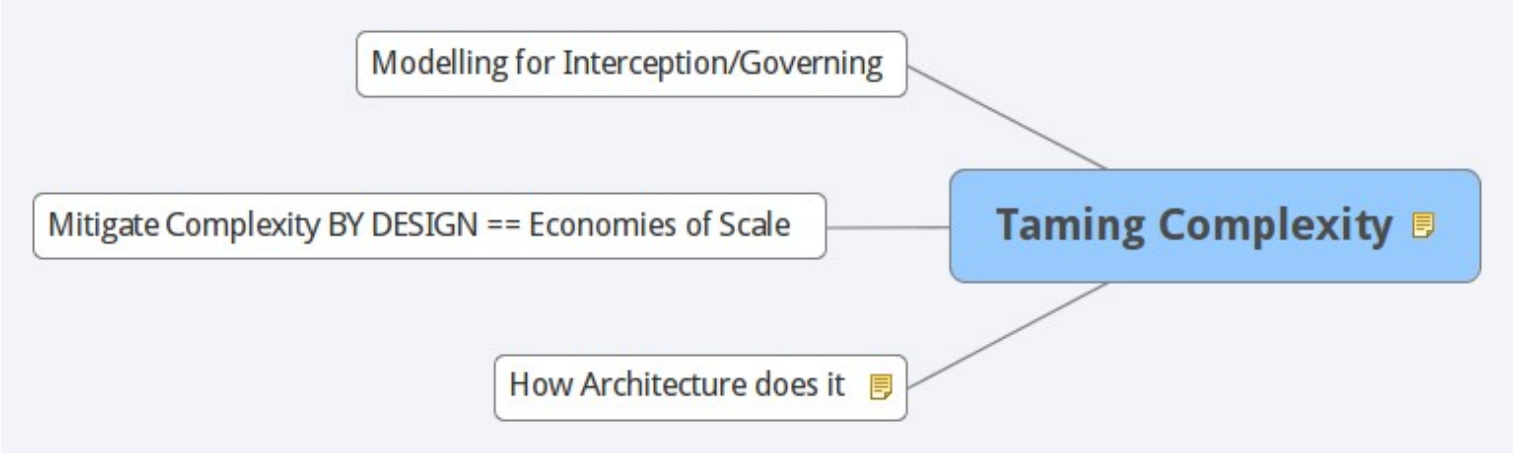


Taming Complexity

Architecture Frameworks provide an answer to "think" Complexity, but not "manage" it.

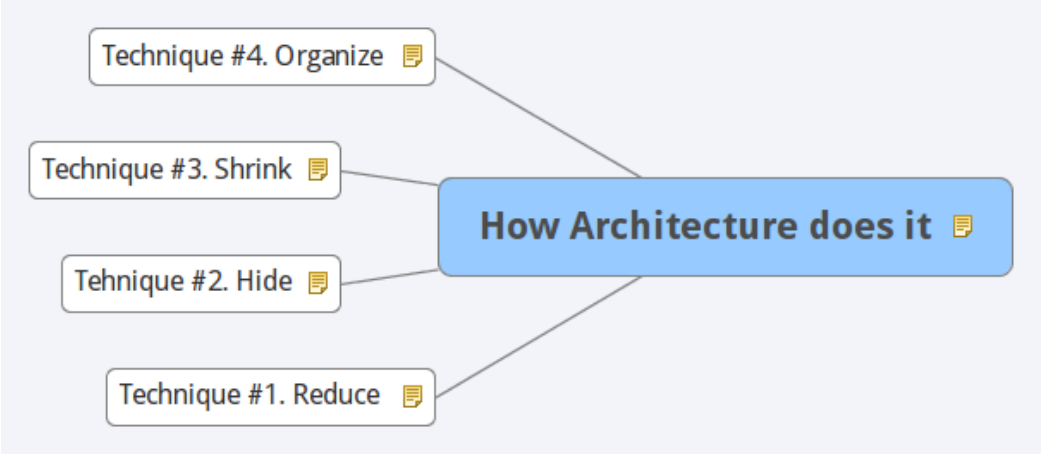
EA Frameworks are used to: (1.) identify what matters to a System (i.e. significant Architectural concerns), (2.) determinate arbitrary limitations.

Elements that are not significant enough: (1.) can be resolved later down the road, (2.) or dismissed altogether, by CMM-compliant processes.



How Architecture does it

EA Frameworks provide Techniques to "think" complexity by:
(1.) "Reducing" the number of kinds of things,
(2.) "Hiding" (removing) elements from select viewpoints,
(3.) Shrinking (exposing) to a simplified View,
(4.) Organizing (sometimes imposing) a Pattern.



Technique #1. Reduce

"Reduction" is performed via divide & conquer approaches (using Views, Viewpoints, Tactics.

It decomposes Systems in small discrete components (using intersecting Views) implicitly building upon one another.

Tehnique #2. Hide

"Hiding" is performed via design practices (Clear partition of concerns, Right level of Abstraction (via object-oriented modeling), Modularity, Encapsulation).

It orchestrates the way Components communicate using Processes and Service interfaces to explicitly forge relations.

Technique #3. Shrink

"Shrinking" (exposing) simplified views via different Types of Architectures: (Business, Data, Application, Technology).

Technique #4. Organize

"Organizing" is performed via best practices for inductive reasoning: Patterns, Blueprints, Principles, Formalization of Artifacts, Ontologies, Modeling, Representation etc.