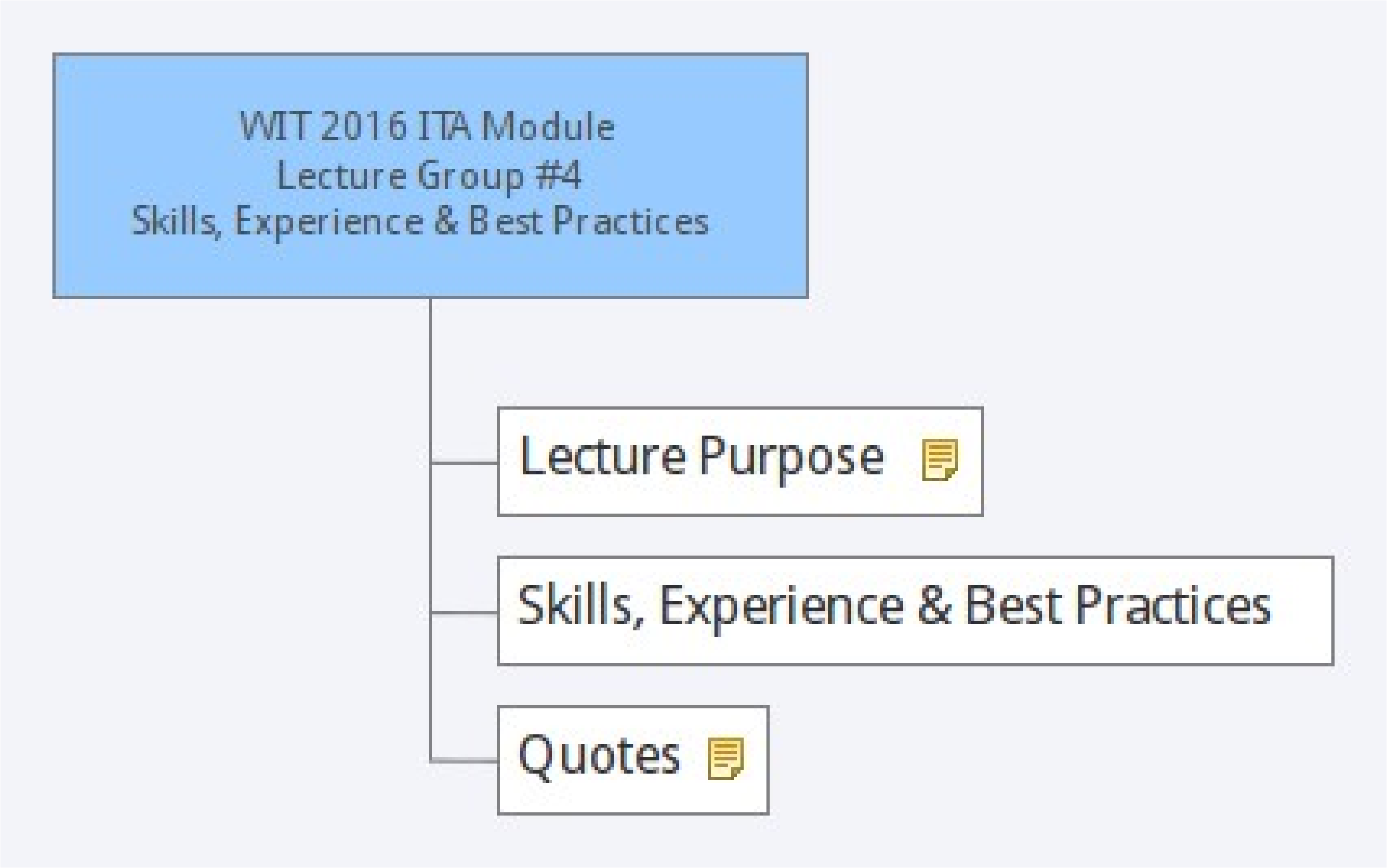
WIT 2016 ITA Module

Skills, Experience & Best Practices
Lecture Group #4

Lecture Group #4
Skills, Experience & Best Practices



WIT 2016 ITA Module
Lecture Group #4
Skills, Experience & Best Practices

- Lecture Purpose
- Skills, Experience & Best Practices
- Quotes

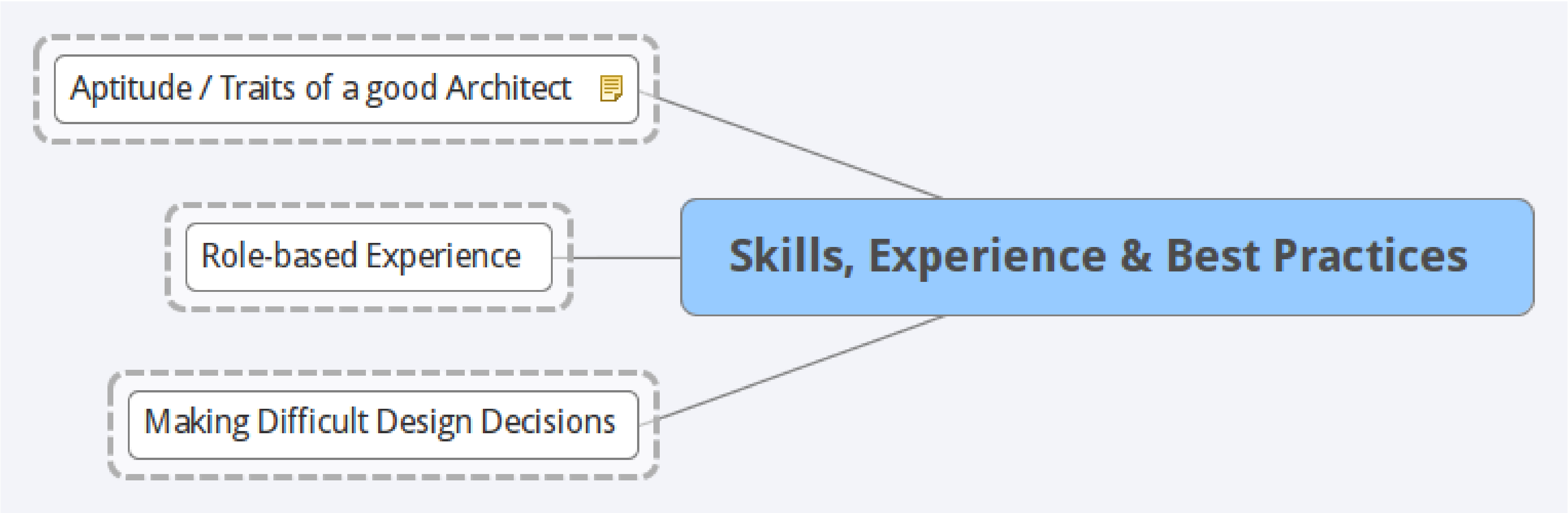
Lecture Purpose

- About sharing grounded experiences in Architecting Enterprise Solutions

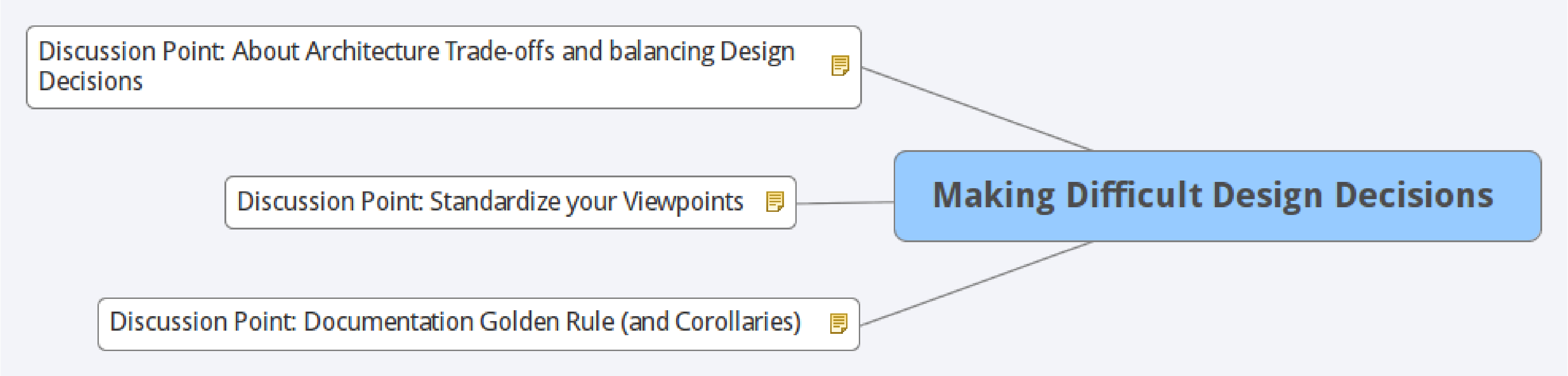
- About sharing common mistakes and best practices, ...other kind of advices from the field.

This lecture is about maximizing chances of success in creating a good project assignment.

Skills, Experience & Best Practices


Aptitude / Traits of a good Architect
Role-based Experience
Making Difficult Design Decisions
Skills, Experience & Best Practices

Making Difficult Design Decisions



Making Difficult Design Decisions

Discussion Point: About Architecture Trade-offs and balancing Design Decisions

Discussion Point: Standardize your Viewpoints

Discussion Point: Documentation Golden Rule (and Corollaries)

Discussion Point: Documentation Golden Rule (and Corollaries)

Golden rule:

- Only provide people with information that they can practically use (or request information that they can practically provide).

Corollary is:

- You will not be able to support all the documentation needs of the organisation with a single model.

Corollary of that is:

- If you want prevent a chaos of incoherent documentations, you need to set up a coherent set of them.

Discussion Point: Standardize your Viewpoints

Standardized Viewpoints are essential for Architects to:

- Help stakeholders understanding risk / cost, and the possibilities of the proposed architecture

- Help decision making and to record design decisions and trade-offs made in the solution design over time

- To avoid the "short memory syndrome", because easy to recover, re-understand, remember the reasons behind design choices.

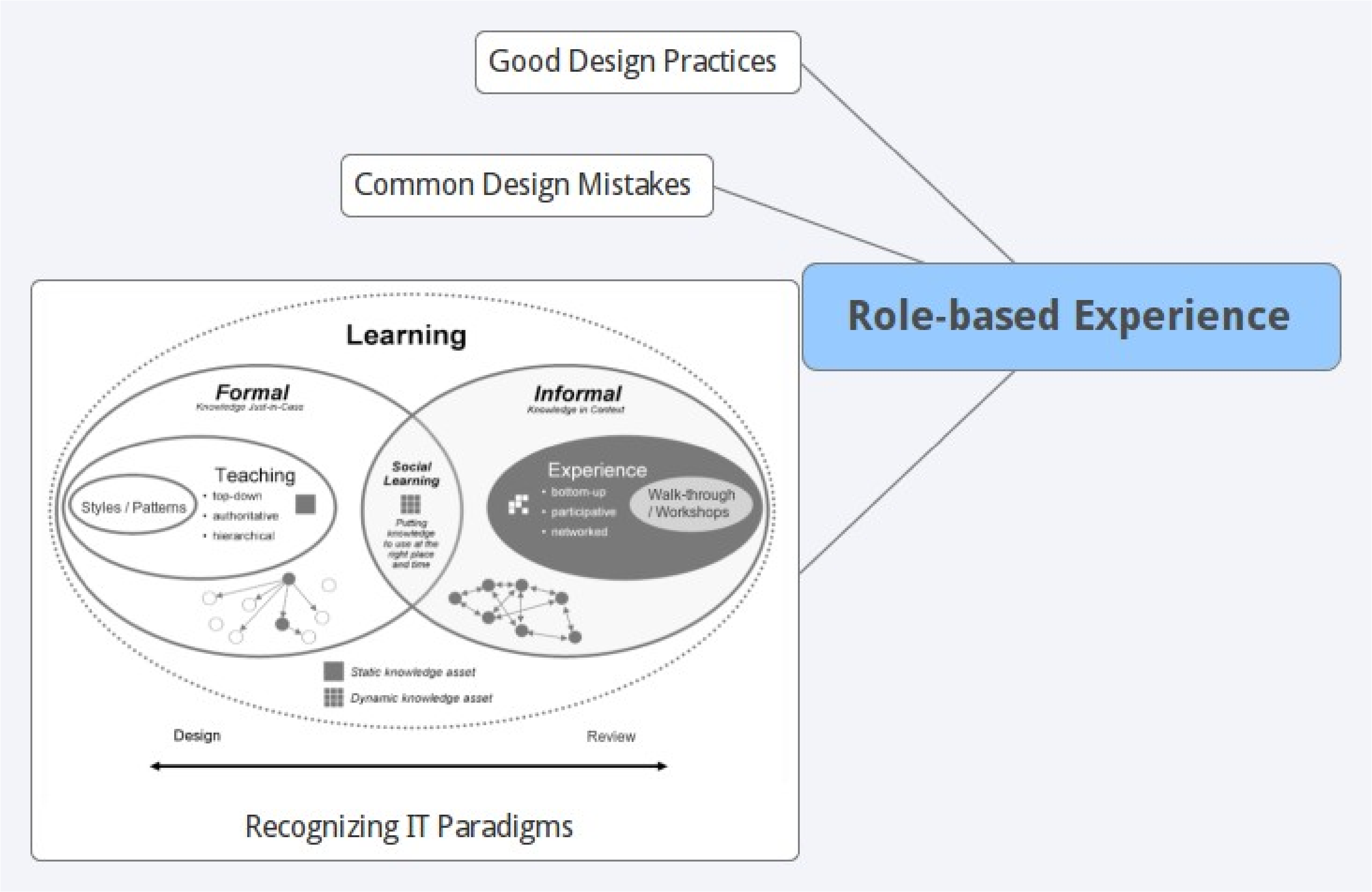
Discussion Point: About Architecture Trade-offs and balancing Design Decisions

- Review some Trade-offs for Performance perspective: Performance and Architecture Styles

- Review some Trade-offs for Availability perspective: Availability and Architecture Styles

- Review some Trade-offs for Scalability perspective: Scalability and Architecture Styles

Role-based Experience



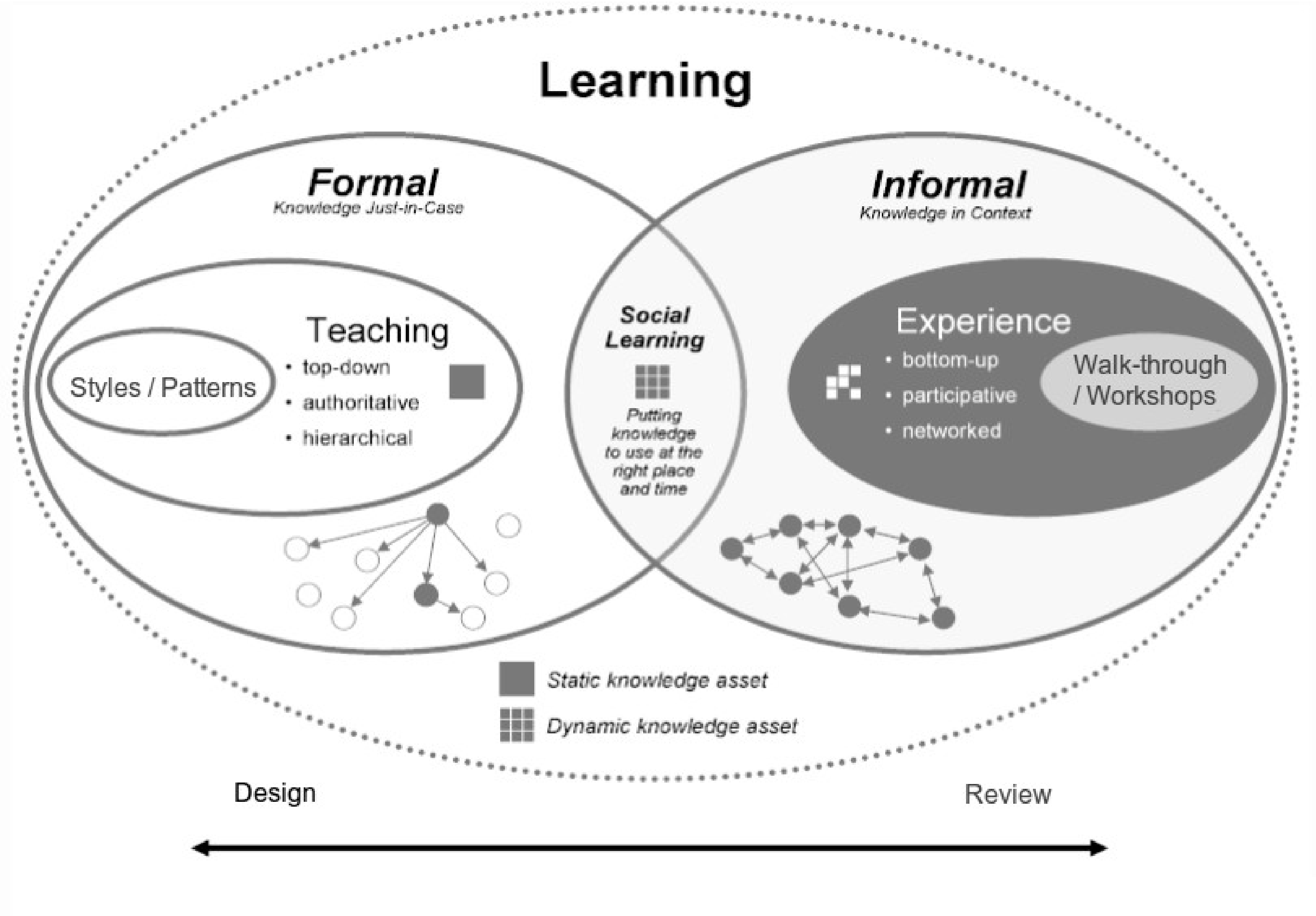
Role-based Experience

Good Design Practices

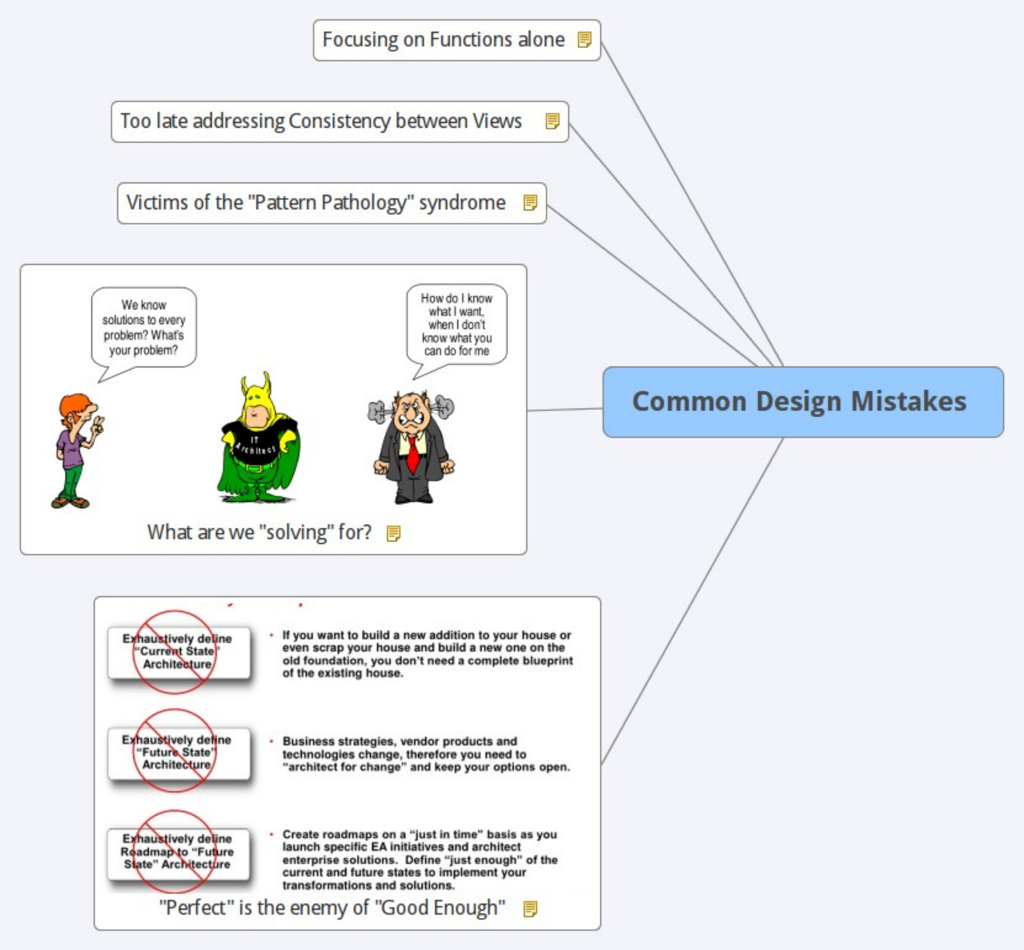
Common Design Mistakes

Recognizing IT Paradigms

Recognizing IT Paradigms

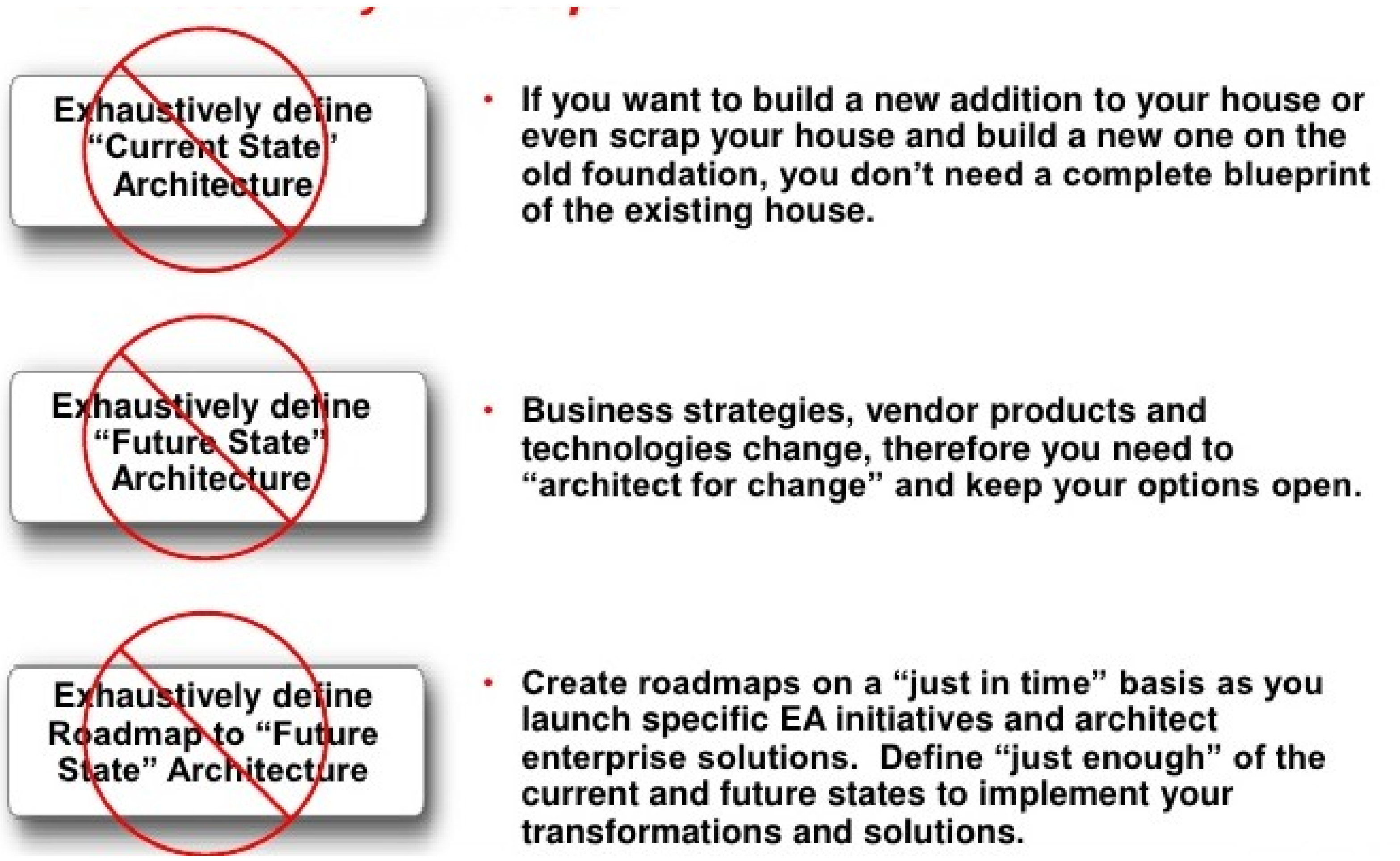


Common Design Mistakes


Common Design Mistakes
Focusing on Functions alone
Too late addressing Consistency between Views
Victims of the "Pattern Pathology" syndrome
We know solutions to every problem? What's your problem?
How do I know what I want, when I don't know what you can do for me
IT Architect
What are we "solving" for?
Exhaustively define "Current State" Architecture
Exhaustively define "Future State" Architecture
Exhaustively define Roadmap to "Future State" Architecture
"Perfect" is the enemy of "Good Enough"

- If you want to build a new addition to your house or even scrap your house and build a new one on the old foundation, you don't need a complete blueprint of the existing house.
- Business strategies, vendor products and technologies change, therefore you need to "architect for change" and keep your options open.
- Create roadmaps on a "just in time" basis as you launch specific EA initiatives and architect enterprise solutions. Define "just enough" of the current and future states to implement your transformations and solutions.

"Perfect" is the enemy of "Good Enough"



Exhaustively define "Current State" Architecture

- If you want to build a new addition to your house or even scrap your house and build a new one on the old foundation, you don't need a complete blueprint of the existing house.

Exhaustively define "Future State" Architecture

- Business strategies, vendor products and technologies change, therefore you need to "architect for change" and keep your options open.

Exhaustively define Roadmap to "Future State" Architecture

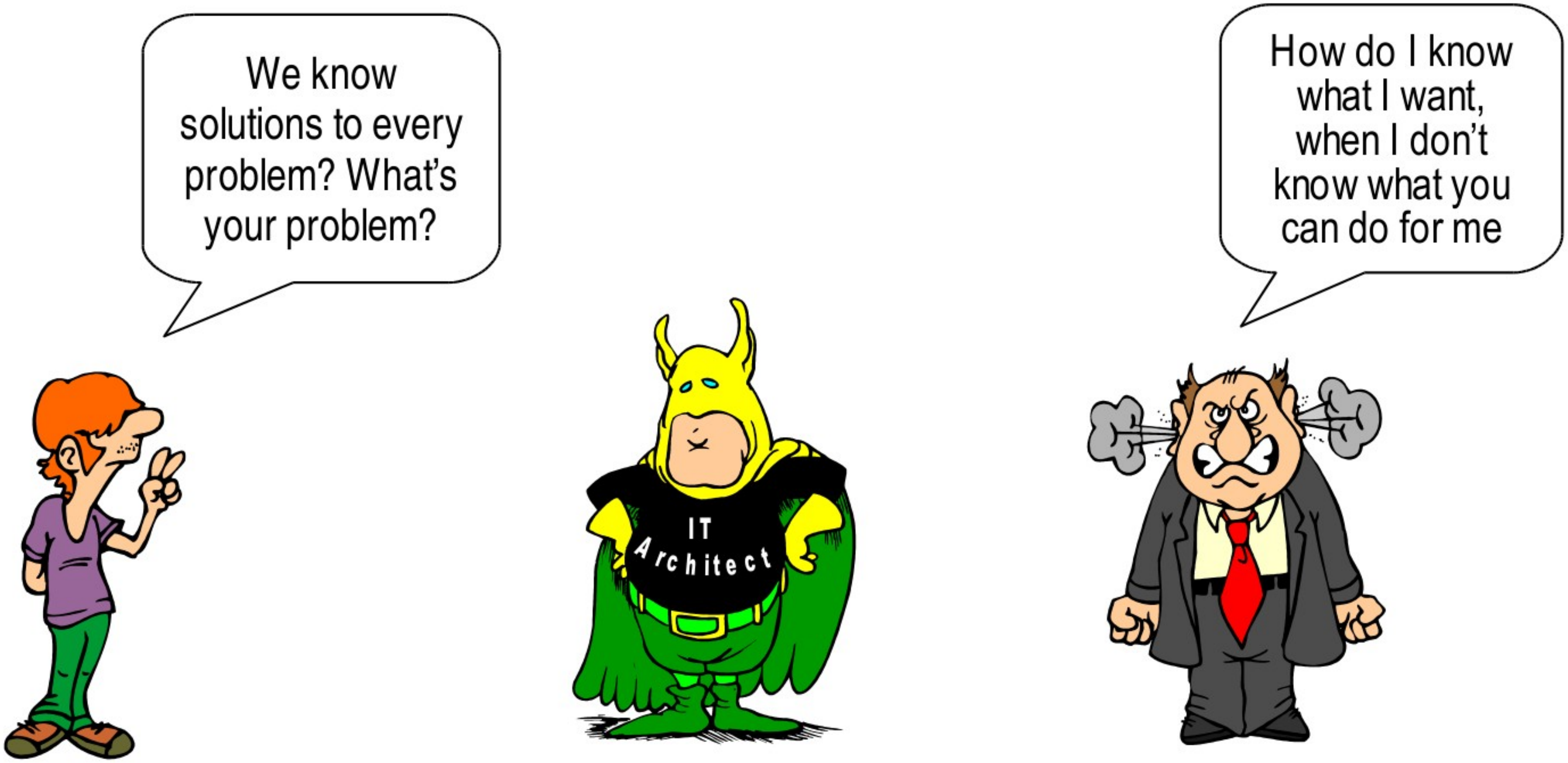
- Create roadmaps on a "just in time" basis as you launch specific EA initiatives and architect enterprise solutions. Define "just enough" of the current and future states to implement your transformations and solutions.

"Perfect" is the enemy of "Good Enough"

Using ambiguous Box and Line Descriptions.

Ambitious Scope, Ambitious Solution Concept, ...forgetting that it Needs to be Built.

What are we "solving" for?


We know solutions to every problem? What's your problem?
IT Architect
How do I know what I want, when I don't know what you can do for me

What are we "solving" for?

While an excellent question: "What are we solving for is a double edge" technique.

No everything is Architect is about solving. "Optimization" an Architectural Solution often is. Inventing a "Creative" Architecture often isn't.

The architect acts to translate between the problem domain concepts of the client and the solution domain concepts of the builder.

Great architects go beyond the role of intermediary to make a visionary combination of technology and purpose that exceeds the expectation of the builder or client.

Victims of the "Pattern Pathology" syndrome

Good Architects validate the fitness for purpose of the Style, Pattern, View, Artifacts, Deliverable they use.

As such, they always are in a position to explain WHY a Style or Pattern is used.

"Without changing our patterns of thought we will not be able to solve the problems we created with our current patterns of thought".

Too late addressing Consistency between Views

It is no good waiting until your models are nearly complete to determine whether they are consistent with one another.

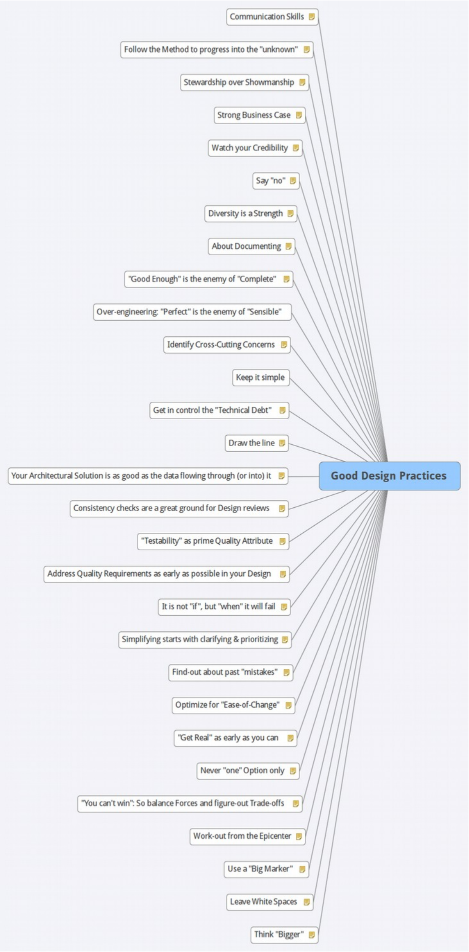
Not doing consistency checks early induces: (1.) Re-work, (2.) Additional reviews, and (3.) potential confusion, mis-communications, mis-representations by Stakeholders.

Focusing on Functions alone

An Architecture only answering the Functional Requirements is only 50% accomplishing half of its mandate.

While in no position to express them as requirements, Users are extremely sensitive to Security, Performance, Availability.

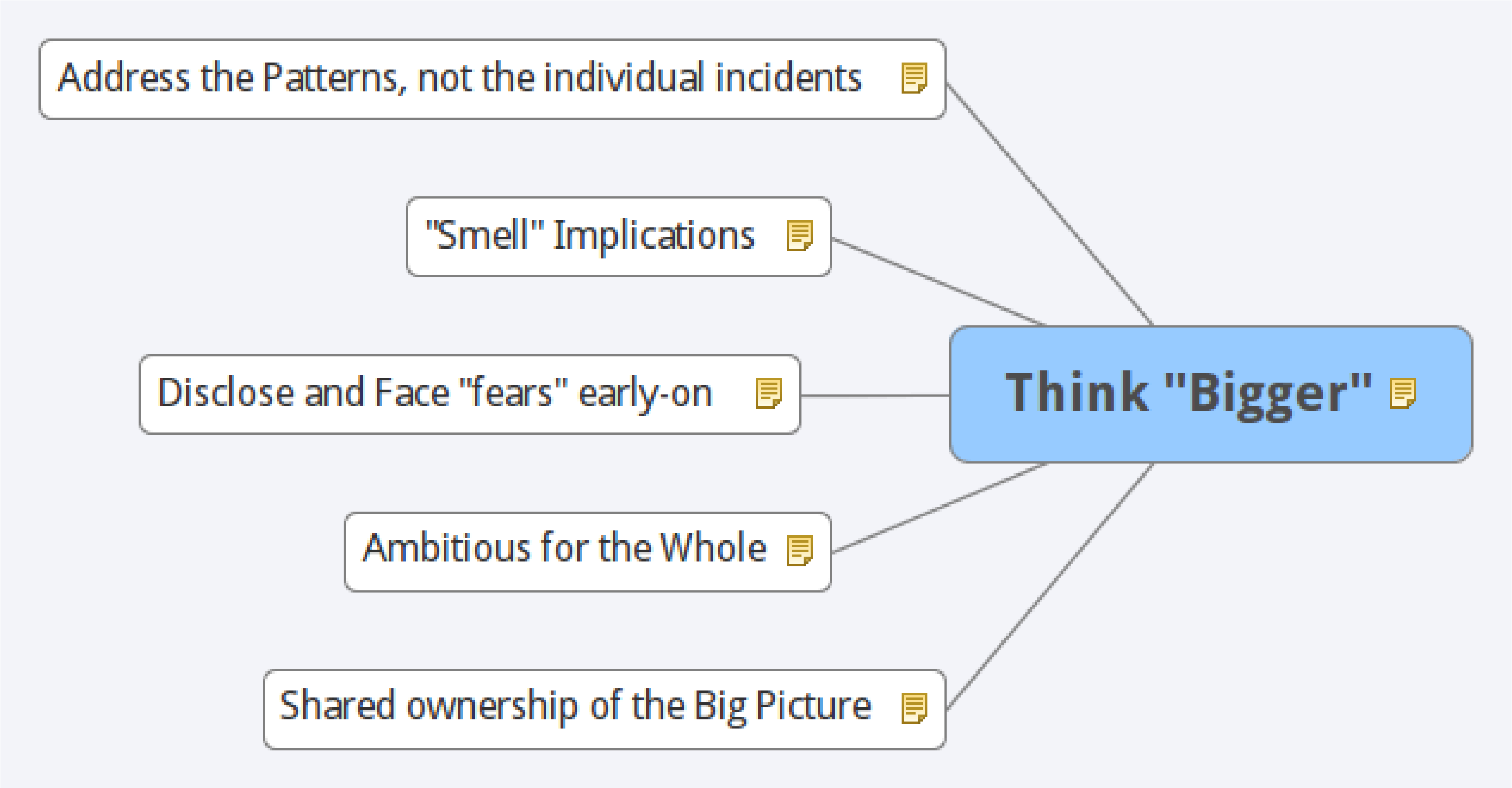
Good Design Practices



Good Design Practices

- Communication Skills
- Follow the Method to progress into the "unknown"
- Stewardship over Showmanship
- Strong Business Case
- Watch your Credibility
- Say "no"
- Diversity is a Strength
- About Documenting
- "Good Enough" is the enemy of "Complete"
- Over-engineering: "Perfect" is the enemy of "Sensible"
- Identify Cross-Cutting Concerns
- Keep it simple
- Get in control the "Technical Debt"
- Draw the line
- Your Architectural Solution is as good as the data flowing through (or into) it
- Consistency checks are a great ground for Design reviews
- "Testability" as prime Quality Attribute
- Address Quality Requirements as early as possible in your Design
- It is not "if", but "when" it will fail
- Simplifying starts with clarifying & prioritizing
- Find-out about past "mistakes"
- Optimize for "Ease-of-Change"
- "Get Real" as early as you can
- Never "one" Option only
- "You can't win": So balance Forces and figure-out Trade-offs
- Work-out from the Epicenter
- Use a "Big Marker"
- Leave White Spaces
- Think "Bigger"

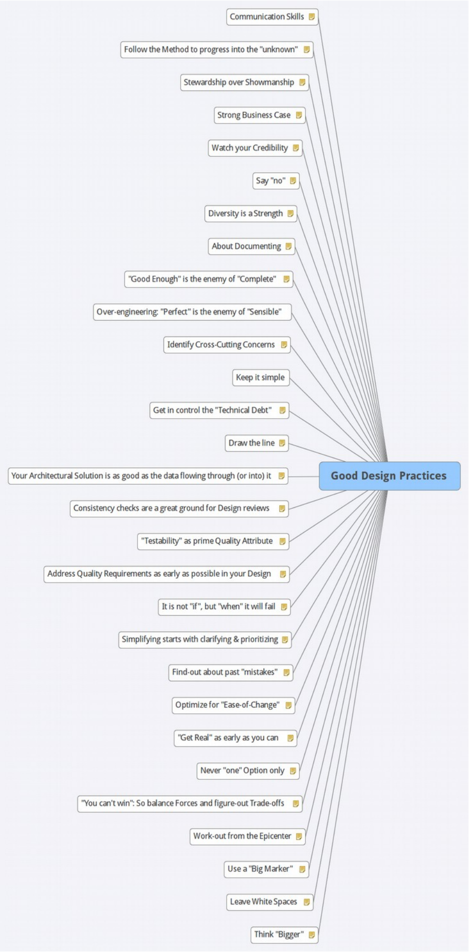
Think "Bigger"



Think "Bigger"

- Address the Patterns, not the individual incidents
- "Smell" Implications
- Disclose and Face "fears" early-on
- Ambitious for the Whole
- Shared ownership of the Big Picture

Good Design Practices



Good Design Practices

- Communication Skills
- Follow the Method to progress into the "unknown"
- Stewardship over Showmanship
- Strong Business Case
- Watch your Credibility
- Say "no"
- Diversity is a Strength
- About Documenting
- "Good Enough" is the enemy of "Complete"
- Over-engineering: "Perfect" is the enemy of "Sensible"
- Identify Cross-Cutting Concerns
- Keep it simple
- Get in control the "Technical Debt"
- Draw the line
- Your Architectural Solution is as good as the data flowing through (or into) it
- Consistency checks are a great ground for Design reviews
- "Testability" as prime Quality Attribute
- Address Quality Requirements as early as possible in your Design
- It is not "if", but "when" it will fail
- Simplifying starts with clarifying & prioritizing
- Find-out about past "mistakes"
- Optimize for "Ease-of-Change"
- "Get Real" as early as you can
- Never "one" Option only
- "You can't win": So balance Forces and figure-out Trade-offs
- Work-out from the Epicenter
- Use a "Big Marker"
- Leave White Spaces
- Think "Bigger"

Shared ownership of the Big Picture

Architects think big and coach your staff to think bigger.

Start conversations with the grander vision that everyone buys into - e.g. The bottom line: greater productivity... more sales, etc.

People are likely to feel less defensive and more willing to change attitudes or behaviours when they are guided to see how their understanding and alignment to the big-picture impacts on the wider organisation.

Ambitious for the Whole

Architects know that Solutions work best when the egos of "experts" are not getting in the way.

This means they are able to be ambitious for the System and the Team building it.

Such Architects are highly ambitious, but they exercise this with great humility.

Disclose and Face "fears" early-on

Architects are not afraid to ask & answer very early: “What does a failure of delivered Architectural Solution would imply for the Team, the Enterprise?”.

They see ‘failure’ as a learning opportunity and to look at these as just a bunch of **really** useful information.

"Smell" Implications

Architects know that imbalances in one part of their Solution will send ripples to other areas of the business.

Tense working relationships can spread their effect like a virus.

Think bigger when there is a strain in a relationship in your workplace. Try asking yourself who ELSE this will affect - i.e. "Who should I involve?".

In effect, write down the names of the key people involved in the situation. Under each of the names write down 3 names of people who are directly linked to that person who will also be affected by the situation.

Address the Patterns, not the individual incidents

Whenever a difficult situation arises, it is important not to see it as a one-off event.

Think bigger and recall similar situations that might have occurred and how often.

Leave White Spaces

Architects must not, and are not expected to answer every aspect of a Solution Architecture.

On the contrary, they leave "white space" for Subject Matter Experts to fill.

As contributions add-up, so is the shared accountability for the end result.

Use a "Big Marker"

In the 1st passes of Architecture Design, use a "big marker" to draw what matters.

Good Enterprise Architects are capable of going deep because they care about their field...

...but it is important to avoid doing it, and get trapped in the details...

...details that will change down the line anyway.

Build in and maintain options as long as possible in the design and implementation of complex systems. You will need them.

Work-out from the Epicenter

Architects determine early what is the most important requirement the Architectural Solution must answer to.

...then exercise the Design answering this requirement.

The resulting experience of this simulation should drive the next step of the Design process.

Architects keep working out from the center until they are done.

"You can't win": So balance Forces and figure-out Trade-offs

An Architecture can rarely answer all the concurrent requirements a Business Initiative brings.

A lot of work goes into balancing forces, by forcing prioritization and outlining Requirements pulling in opposite directions.

A lot of creativity goes into figuring-out Trade-offs answering to some satisfying the constraints exercised on the Architecture.

Never "one" Option only

One size does not fit all; that is, there may be more than one Architecture relevant for a System.

Prepare to present at least 2 Architectural Options for the Solution Architecture.

If there is only 1 solution, get a second opinion.

Presenting Options helps your audience to better value the amount of work and did go into establishing an Architectural Recommendation.

Presenting Options demonstrates how the Enterprise makes decision. With each decision comes buy-in (hence a part of responsibility) on the agreed Architectural Solution.

"Get Real" as early as you can

"If you build a house, build it in card board first. You never know if a building is good until you come inside" or until you're using it.

Build something cheap fast and then experience it.

Architects try to "get close" to real as possible, early, to see things in Action.

Exercising an Architectural Solution using Scenarios, is one way to "get real" early.

Optimize for "Ease-of-Change"

Great Solutions are not built, but Grown.

Learn from History, Look for Reference, Principles.

Beware of "great" ideas.

Find-out about past "mistakes"

Give as much importance to mistakes performed by other Projects, other Architects, than you do about best practices.

Patterns, Styles, reference Architectures focus on coherence, but strip-out context and environmental factors.

The challenge of an Architect comes from the constraints from its environment.

Simplifying starts with clarifying & prioritizing

Ensure that your Architectural Solution delivers excellently on its Core mission, its Core purpose.

Simplify essential complexity. Prefer a solid design, without functionality gaps, and answering well all the Quality Properties, before getting distracted by non-essential Design aspects.

Your biggest problem isn't always Technical. Maximize the work NOT done to disentangle and diminish accidental complexity.

Cut the unnecessary. In general, be jealous of your energy and time to the "significant" design decisions.

When not immediately significant, defer design decisions until timing is for decision is more appropriate (e.g. more information may be available).

It is not "if", but "when" it will fail

Do not underestimate how frequently your Solution Architecture will "fail". Because it will.

You need to account for various degrees of failure... and recovery.

Building contingency plans involve close collaboration with a number of subject matter experts from Business (BCP) and Technology (DR) alike.

Address Quality Requirements as early as possible in your Design

Applying Quality Properties to a Solution Architecture after the fact does not work.

Perspectives relating to Performance, Availability, and Resilience have to be designed into your solution from the start.

"Testability" as prime Quality Attribute

There is hardly any excuse for an Architect to avoid specifying the "testability" of its Solution Architecture.

"Testability" occurs are different levels and is not (cannot) be the sole accountability of QA Functions any longer.

A number of Quality Properties will directly benefit from it, including load testing, performance testing, security testing, Other.

Consistency checks are a great ground for Design reviews

Architectural Description (AD) Deliverables greatly benefit from early review of Consistency checks.

Consistency checks are a great, natural and intuitive Agenda for Architectural Workshops grouping Stakeholders and Subject Matter Experts.

With each Gap of "inconsistency" discovered, comes a feeling of achievement and a better management of the risk.

Internal consistency compare and connects models in a same View, and between views of a same Architectural Domain.

External consistency compare and connects Views between Architectural Domains.

Your Architectural Solution is as good as the data flowing through (or into) it

The line between Application Architecture and Information Architecture clarifies, but neither Architects from each Domain can afford to ONLY know their Domain.

Accountability on how Data is managed by the Solution Architecture is a shared by both roles.

Great content creates great solutions.

This is increasingly true when building SOA and Cloud Solution Architectures.

Draw the line

Don't micro-manage, but try to "risk manage" instead.

The resulting Implementation will never look exactly the way you designed it.

Give credit to the amendments designers and implementers suggest down the line.

However, do show interest and show your are "clued-in" to the aspects of the Implementation.

Get in control the "Technical Debt"

An IT Architecture is legacy the minute it is checked-in in Production.

As a result, an Architect needs to Design for it, and provide contingency plans.

This concerns, Solution enhancements integrating new requirements, longevity of the Technology stack, and Other considerations.

Focus on Application support and Maintenance implications.

Keep it simple

- Speaks for itself

Identify Cross-Cutting Concerns

After you define the layers, you must identify the functionality that spans layers. This functionality is often described as crosscutting concerns, and includes logging, caching, validation, authentication, and exception management. It is important to identify each of the crosscutting concerns in your application, and design separate components to manage these concerns where possible. This approach helps you to achieve of better reusability and maintainability.

Avoid mixing the crosscutting code with code in the components of each layer, so that the layers and their components only make calls to the crosscutting components when they must carry out an action such as logging, caching, or authentication. As the functionality must be available across layers, you must deploy crosscutting components in such a way that they are accessible to all the layers—even when the layers are located on separate physical tiers.

"Good Enough" is the enemy of "Complete"

Scoping doesn't map to Architectural Increments, leading to release integrity issues.

Disaster Recovery and Business Continuity as an afterthought.

Using ambiguous Box and Line Descriptions.

Ambitious Scope, Ambitious Solution Concept, ...forgetting that it Needs to be Built.

About Documenting

Use a standard and recurrent layout & organization, but make sure that you know what question(s) are being answered by each section of a documented Deliverable.

Doing so will help to unambiguously record Reasons and rationale for Architectural Decisions.

Eliminate whole classes of linguistic ambiguity from a Work-Product - have precise enough semantics to be unambiguous.

Always explain your intent and notation.

Diversity is a Strength

Architects know that a System needs a range of different parts to achieve its purpose.

This means they value people, for (and not) despite their differences.

Architects need diverse styles and opinions to add value to proposed Architectural Solutions.

Ideally Architects work at forming a culture of shared accountability and collaboration.

Say "no"

Architects are often expected to be the "curators" and say 'no'.

An Architectural Solution cannot do it all, neither it should do it all.

Watch your Credibility

Credibility is a key and very important aspect of the successful Architect.

It helps to: (1.) Assert her influence, (2.) Bring Stakeholders to a consensus, (3.) "Hold the line" and keep Stakeholders to their commitments.

Strong Business Case

Ensure a Business Infrastructure is in place.

Usage == Value!

For the User, the interface IS the System.

Stewardship over Showmanship

Think bigger and view your team or organisation as a unit, rather than a collection of individuals, then ask yourself, “How can I catalyse all the resources within this unit to the best advantage of the whole”?

Follow the Method to progress into the "unknown"

Methods provide structure to our thought processes to give us an idea of “the next thing to do”.

Once you let go of the fear of not having “the next thing to do” the structures created by your use of method, in your mental “muscle memory” will take over.

You will come to understand that the structure of the method which once empowered you feels restrictive.

This is the breakthrough point at which your creativity is freed.

But, you really have not left method behind, it still informs what you do.

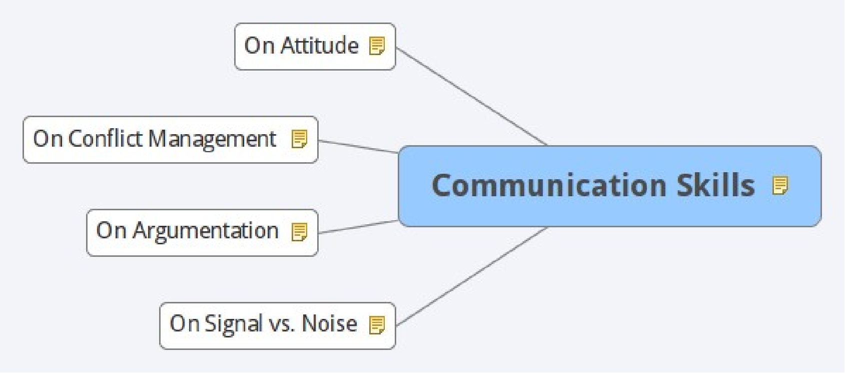
Communication Skills

In Enterprise Architecture, "Communication" goes beyond clarity of elocution and inter-personal skills.

It is about fidelity of reformulation, restitution.

It is about questioning, timing, influencing.

It is about clearing the way, challenging egos, challenging assumptions, provide air-cover and foster a creative culture of resolution.



On Signal vs. Noise

When individuals process, they encode information in Mental Models. When individuals rehearse, they had a bit of themselves based on the Mental Model they restitute.

The worst a Mental Models, the worst is the restitution - i.e. source of complexity and loss of signal quality.

In the Enterprise, all Employees' primary purpose is to Encode and Decode information. An Architect aims to forge a Mental Model for the Organization.

Too many layers augments loss of signal - i.e. source of unnecessary complexity.

On Argumentation

The vulnerability of an argument lies in its premises rather than its conclusion.

When well formulated premises to an argument conclusion are contested and attacked, advances are easily deflected and rebuffed.

Chip away at enough of its premises and the entire argument will eventually implode, collapsing under its own weight into a pile of nonsense and falsehoods.

On Conflict Management

Architects know how to have difficult conversations with Stakeholders while still maintaining positive working relationships.

Good Architects are able to hold the greater good as paramount, and are open to being proved wrong.

They keep thinking big in conflicting situations, examine their own motivations then try to stand in the other person shoes to think & feel as *they* do.

Doing so, gives an Architect a better intuition on a Stakeholder motives, interests and intentions.

On Attitude

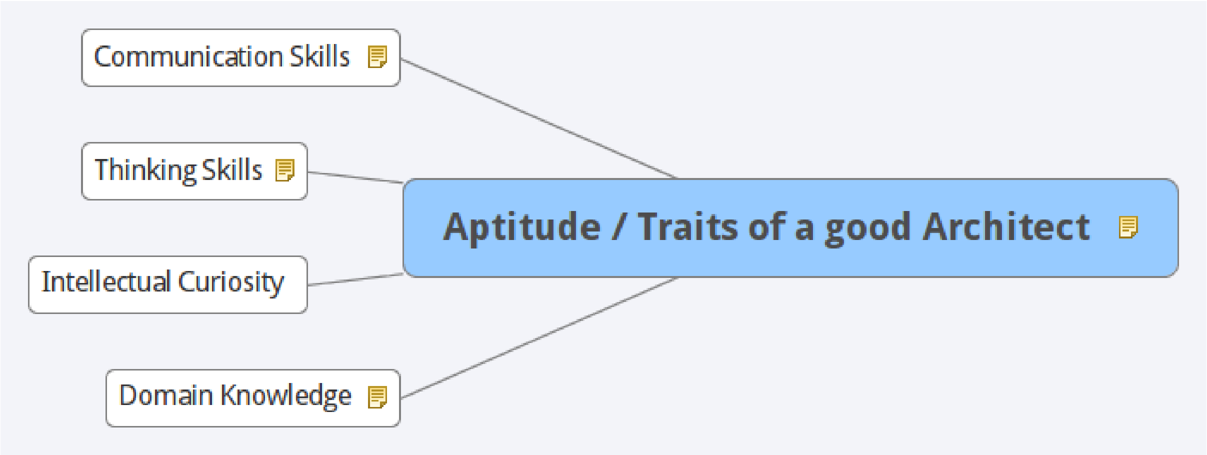
"It's not what you know (IQ), but how you use it (EQ)".

An Architect understands and appreciates the impact of "change", especially what it means for People.

She easily forgets about the glory of resolving alone, will respect Subject Matter Experts - e.g. learn from older Technologies & more experienced folks.

Aptitude / Traits of a good Architect

A good Architect is one who can use precise concepts in the construction of an architecture specification, abstraction techniques to define the parts of the architecture, thread in domain-specific business needs, and use good engineering skills in the process.



Domain Knowledge 1/2

An Architect knows it is “better” to do the right thing wrong, than the wrong thing right.

"Domain Knowledge" is knowledge in the **Business** Domain in which the Enterprise evolves. Domain Knowledge is not IT knowledge, neither it is knowledge of a "Technical" nature.

Domain Knowledge is an essential skill for an Enterprise Architect, and greatly helps to better perform in her role.

Domain Knowledge helps to appreciate the world from different angles.

Domain Knowledge is a terrific weapon to "maximize what not to do", challenge assumptions, ask uncomfortable questions about the intrinsic value of some of the requested features.

Domain Knowledge helps to understand the culture of the industry, the culture of the Enterprise and hence the bigger picture and implications of change.

Domain Knowledge 2/2

An Architect knows it is “better” to do the right thing wrong, than the wrong thing right.

"Domain Knowledge" is knowledge in the **Business** Domain in which the Enterprise evolves.

It is not IT knowledge, neither it is knowledge of a "Technical" nature.

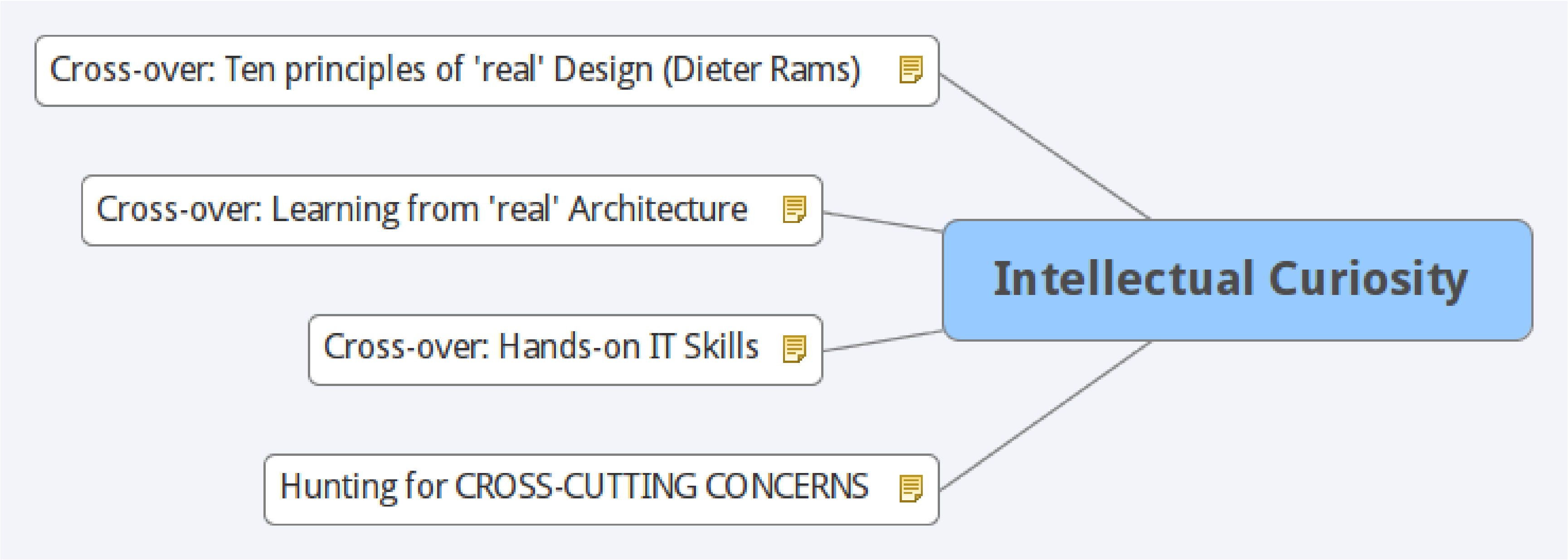
- helps to appreciate the world from different angles.

It is a weapon to "maximize what not to do", challenge assumptions, ask uncomfortable questions about the intrinsic value of some of the requested features.

- helps to understand the culture of the industry, the culture of the Enterprise and hence the bigger picture and implications of change.

It is an essential skill for an Enterprise Architect, and greatly helps to better perform in her role.

Intellectual Curiosity



Cross-over: Ten principles of 'real' Design (Dieter Rams)

Cross-over: Learning from 'real' Architecture

Cross-over: Hands-on IT Skills

Hunting for CROSS-CUTTING CONCERNS

Intellectual Curiosity

Hunting for CROSS-CUTTING CONCERNS

Avoid creating stovepipe application solutions that do not evolve easily.

Instead, work hard at finding cross cutting concerns between a net-new solution and other existing applications.

Cross-over: Hands-on IT Skills

Credibility with Technologists - and any Subject Matter "Experts" in general - is paramount.

Ability to deal across Specializations is Key to the Role.

Broad Technical Knowledge is Valuable, especially of Open Standards.

Experiment and Try before influencing.

Cross-over: Learning from 'real' Architecture

Collision of field leads to insights and mind shifts that can be extremely useful to think "out-of-the-box".

Cross-over: Ten principles of 'real' Design (Dieter Rams)

Good design is innovative.

Good design makes a product useful.

Good design is aesthetic.

Good design helps us to understand a product.

Good design is unobtrusive.

Good design is honest.

Good design is long-lasting.

Good design is consequent to the last detail.

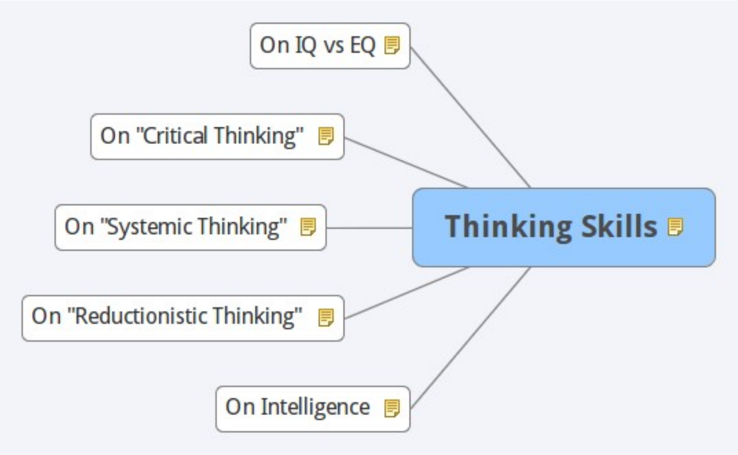
Good design is concerned with the environment.

Good design is as little design as possible.

Thinking Skills

Good Architectural skills and thinking permeate the use of Abstraction and Composition.

Abstraction and composition enable a complex problem to be focused on parts at varying levels of detail.



On Intelligence

Intelligence as IQ: "To represent problems in ways that lead to effective solutions".

Intelligence as EQ: "See the world through the eyes and minds of different experts".

On "Reductionistic Thinking"

Reductionistic Thinking (also known as Analytical Thinking) derives properties of the whole Architectural Solution from the properties of its parts

"Reductionistic Thinking" is a Technique used to understand a complex system, by making sense of what a Solution should be (or is) by breaking it apart.

This type of a analysis focuses is suited for detail-oriented Design activities, focusing on the elements of a System in isolation.

On "Systemic Thinking"

When an Architect designs a house he first sketches the house as a whole and then puts rooms into it. evaluating a room is what effect it has on the whole.

She is willing to make a room worse if doing so will make the house better. She is even willing to make a room worse if doing so will make the house better.

The principal criterion she employs in evaluating a room is what effect it has on the whole.

Systemic Thinking (also called 'Holistic' or 'Synthetic' Thinking) derive properties of parts from properties of the whole that contains them.

On "Critical Thinking"

"Critical Thinking" is the intellectually disciplined process of actively and skillfully conceptualizing, applying, analyzing, synthesizing, and/or evaluating information gathered from, or generated by, observation, experience, reflection, reasoning, or communication, as a guide to belief and action.

A Critical Thinker: (1.) Raises important questions and problems, formulating them clearly and precisely, (2.) Gathers and Assesses relevant information, using abstract ideas to interpret it effectively, (3.) Comes to well-reasoned conclusions and solutions, testing them against relevant criteria and standards, (4.) Thinks open-mindedly within alternative systems of thought, recognizing and assessing, as need be, their assumptions, implications, and practical consequences, (5.) Communicates effectively with others in figuring out solutions to complex problems, without being unduly influenced by others' thinking on the topic".

On IQ vs EQ

Intelligence as IQ: "To represent problems in ways that lead to effective solutions".

Intelligence as EQ: "See the world through the eyes and minds of different experts".

Communication Skills

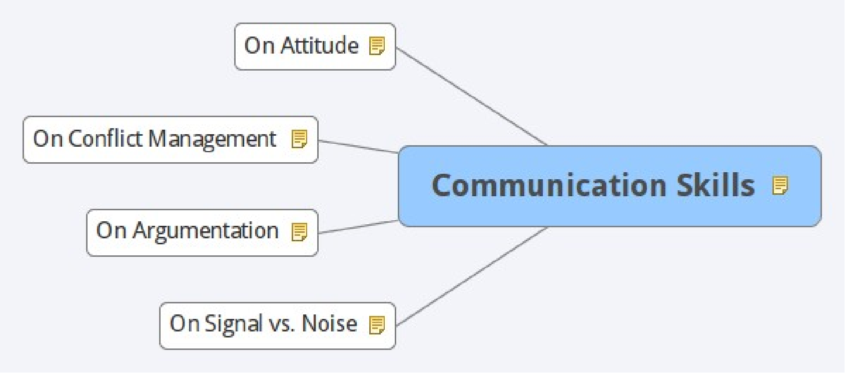
In Enterprise Architecture, "Communication" goes beyond clarity of elocution and inter-personal skills.

It is about fidelity of reformulation, restitution.

It is about questioning, timing, influencing.

It is about clearing the way, challenging egos, challenging assumptions, provide air-cover and foster a creative culture of resolution.

Visit this link: [https://en.wikipedia.org/wiki/How_to_Win_Friends_and_Influence_People#Major_sections_and_points]



On Signal vs. Noise

When individuals process, they encode information in Mental Models. When individuals rehearse, they had a bit of themselves based on the Mental Model they restitute.

The worst a Mental Models, the worst is the restitution - i.e. source of complexity and loss of signal quality.

In the Enterprise, all Employees' primary purpose is to Encode and Decode information. An Architect aims to forge a Mental Model for the Organization.

Too many layers augments loss of signal - i.e. source of unnecessary complexity.

On Argumentation

The vulnerability of an argument lies in its premises rather than its conclusion.

When well formulated premises to an argument conclusion are contested and attacked, advances are easily deflected and rebuffed.

Chip away at enough of its premises and the entire argument will eventually implode, collapsing under its own weight into a pile of nonsense and falsehoods.

On Conflict Management

Architects know how to have difficult conversations with Stakeholders while still maintaining positive working relationships.

Good Architects are able to hold the greater good as paramount, and are open to being proved wrong.

They keep thinking big in conflicting situations, examine their own motivations then try to stand in the other person shoes to think & feel as *they* do.

Doing so, gives an Architect a better intuition on a Stakeholder motives, interests and intentions.

On Attitude

"It's not what you know (IQ), but how you use it (EQ)".

An Architect understands and appreciates the impact of "change", especially what it means for People.

She easily forgets about the glory of resolving alone, will respect Subject Matter Experts - e.g. learn from older Technologies & more experienced folks.

Quotes

"I have the right to be wrong, but I have to be precisely wrong" and that will guarantee your survival in Corporations.

"We are willing to kid ourselves and make others believe we're in control of things".

"In big organizations there's always someone who can retroactively prove he warned you of the certainty of failure if what you do goes wrong".