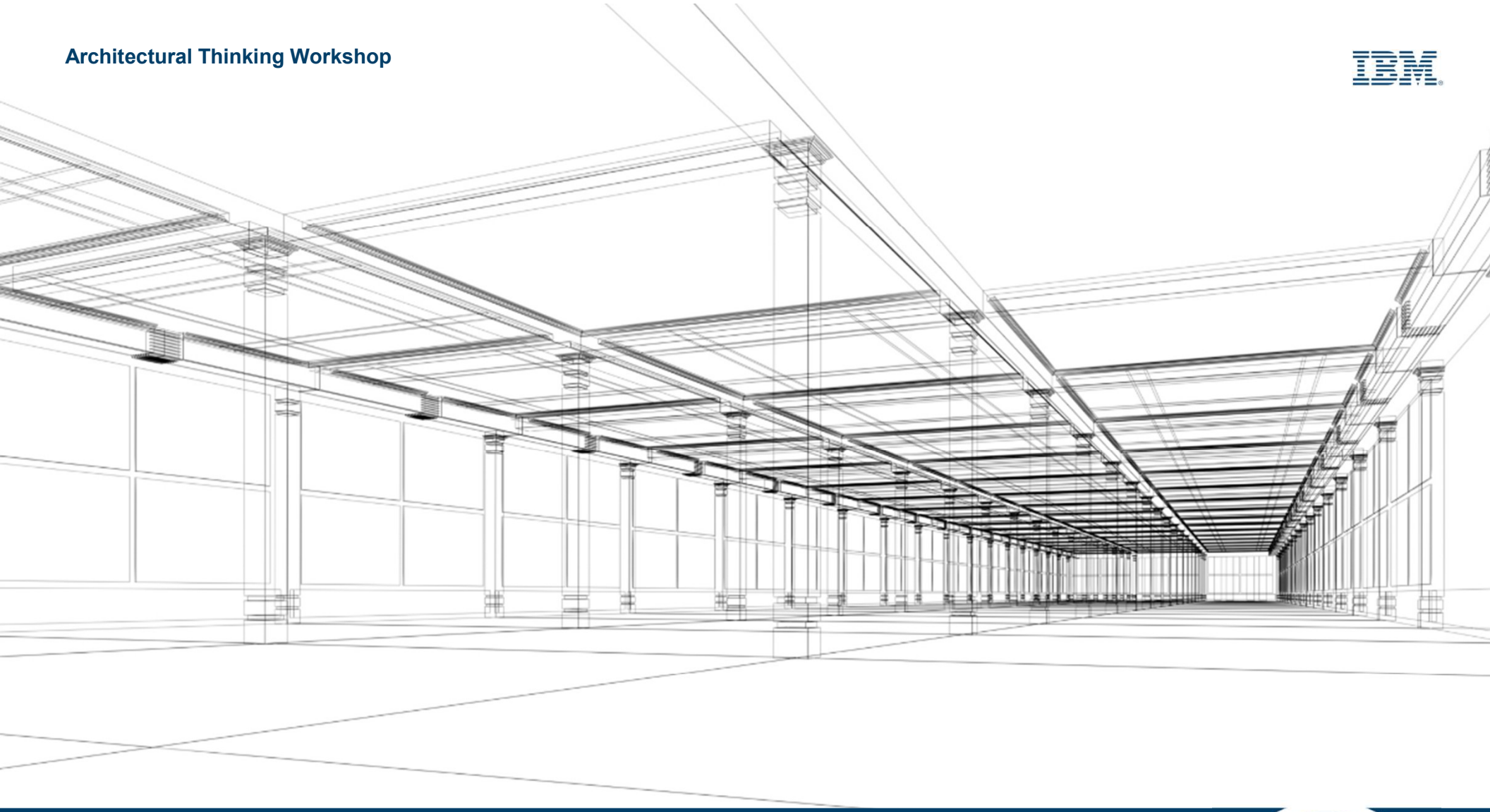


Validation and Viability

Module outline

- **Introduction and objectives**
- Viability
- Validation
- Approaches to viability and validation
- Summary and references

Learning objectives

At the end of this module, you should be able to:

- Describe:
  - Technical reviews and their relationship to the quality assurance and testing processes
  - How to identify resources available to help with viability assessments and technical reviews
  - How to incorporate full life-cycle validation and verification into architectural thinking
- Identify:
  - Those who are concerned about the architecture's quality: the Architect's target audiences and stakeholders
  - How to determine the validity and viability of an architecture
  - How to deal with risk: removal, containment, and acceptance

Why are Viability and Validation important in Architectural Thinking?

- Humans are imperfect and so are solutions created by them. Fixing flawed solutions at an early stage of the project is **much** cheaper than at a later stage.

- Reviews and assessments are used to ensure that the architecture (and its documentation) is ready, ensuring that it:
  - can be implemented and delivered – *Viability*
  - satisfies the functional requirements and qualities – *Validation*

- The Architectural Thinking work product for documenting the architecture assessment is *Viability Assessment*.

- The validation and viability of the architecture are an important element of the overall approach to testing across the whole solution life cycle.

Common causes of troubled projects

Project design

- Fixed price for combined phases
- Vague or misunderstood completion criteria
- Customer responsibilities not clearly identified and documented
- Assumptions not clearly identified and documented
- Concessions during negotiations
- Pricing concessions, “low-balling”
- Poorly written proposals or SOWs
- Inaccurate project estimates
- Not performing QA reviews
- Lack of transition, marketing to delivery

Project delivery

- Project skills deficiencies
- Lack of, or inadequate, project management
- Not performing QA reviews and action plans
- Lack of line management oversight and support
- Not implementing or exercising change control
- Starting a phase before completing the preceding phase
- Unplanned turnover of key project team members

Summary:

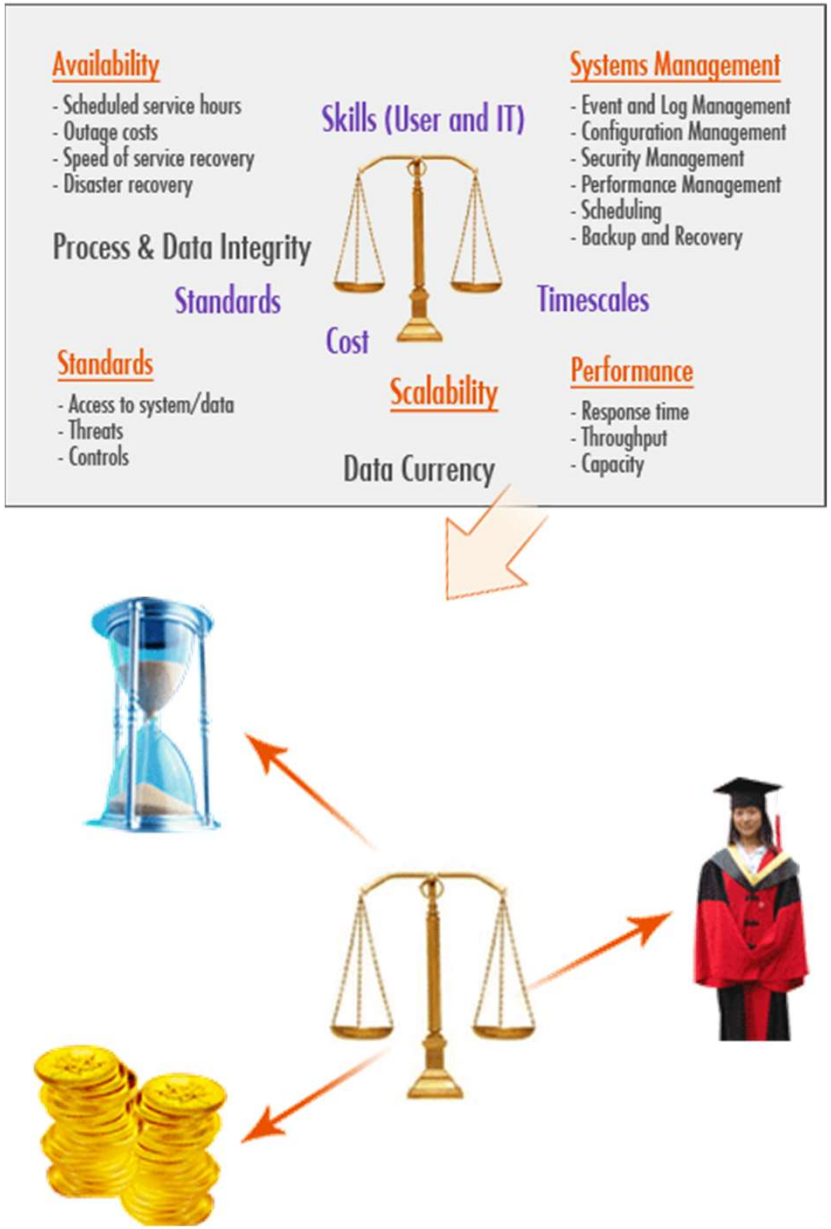
The common causes for troubled projects are known, and almost all are avoidable.

Remember:

- Viability assessments and other forms of validation and QA can help prevent your project from becoming troubled.
- The lists and assessments cannot help unless the recommended actions are taken up. As an Architect, take the lead.

Even most well thought through solution are imperfect

- **Cost:** Budgets are limited. Even if a project starts off well-funded, new cost constraints may cause solution architectures to change radically from initial design.
- **Information:** “As you know more, you know more”. New information: requirements, legislation, competition causes change to the business. The architecture must change to accommodate new information constantly.
- **Skilled resources:** Throughout a long-lived project, staff comes and goes.
- **Time:** Architects never have sufficient time, particularly at the beginning of the project when they have the most influence.



Module outline

- Introduction and objectives
- **Viability**
- Validation
- Approaches to viability and validation
- Summary and references

How to approach the viability assessment

Solution Architects performing the Viability Assessment:

- Assess the architecture. Does it fit the wider context? Does it satisfy the wider needs of the enterprise:
  - As-is operational environment
  - To-be enterprise architecture
- Confirm that the deployed solution will be operable and maintainable with operations
- Understand the context of the architecture. What are the key requirements and commitments?
- Assess whether the proposed solution is possible:
  - Can it be designed and developed within the available means?
  - Will it work? Will it perform? Will it be available?
- Determine the project team's ability to deliver.
  - Are the right resources in place to ensure success?
  - Is the necessary infrastructure in place?
- Identify and assess technical risks.
  - What risks must be raised and formally managed? What are their probable impacts? How can these be contained?

Module outline

- Introduction and objectives
- Viability
- **Validation**
- Approaches to viability and validation
- Summary and references

What is validation?

When examining an architecture's validity, consider the following:

- Does it do what is required of it? Does it meet the sponsor's needs?
- Does it do it well enough? Is it over engineered, or is it vulnerable in some way?

Solution Architects ensure that the proposed solution is acceptable to the various stakeholders:

- The key criterion is that it satisfies the Functional and Nonfunctional requirements
- This is achieved through the use of *Requirements Traceability and Verification Matrix* (RTVM). RTVM is used throughout the whole project, not just at the end

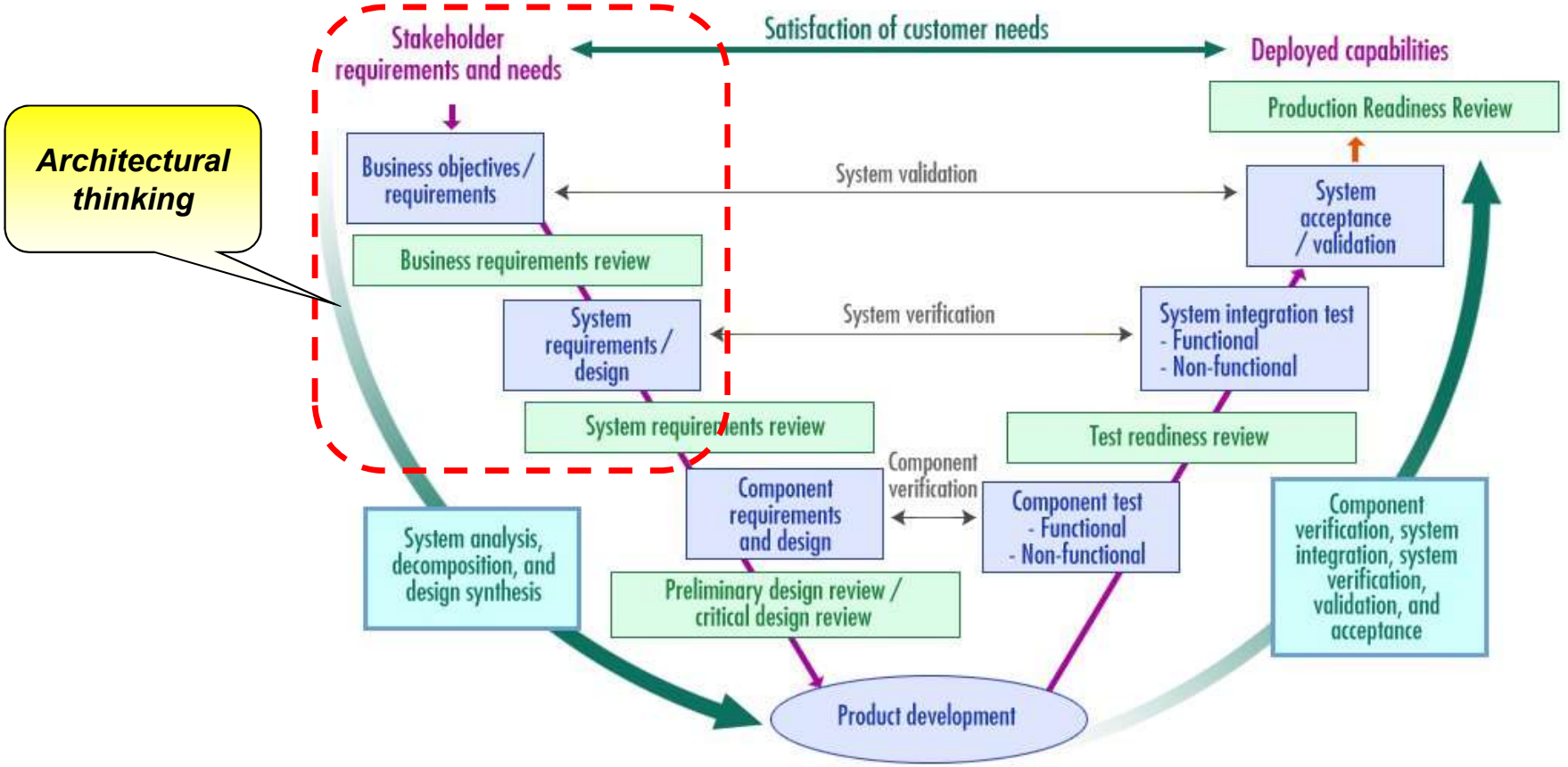
RTVM:

- Ensures completeness of architecture and design through mapping of the requirements (both FRs and NFRs) to the solution components
- Provides the basis for test planning and execution
- Provides a framework for risk management and mitigation
- Provides the basis of clear and timely client communication with regard to schedule, risk, and financial impacts

Validation is not an after the fact concern; it is addressed throughout the project

The testing V model and system engineering baseline reviews are a good starting point for planning validation and verification activities.

The V model ensures that all tests relate to requirements, and all requirements have an associated test case.



Module outline

- Introduction and objectives
- Viability
- Validation
- **Approaches to viability and validation**
- Summary and references

Assessments document problems, findings, risks and recommendations

Are things always what they seem? Two people with the same symptom can have very different illnesses. The same is true with IT architectures. We need a diagnosis and cure.

Problems: *What's wrong? Let's do an examination.*

- Symptoms, direct observations about the IT architecture

Findings: *Based on what I've seen, I think you've got….*

- One or more symptoms will indicate a deeper cause - statements of facts (or assertions) about the architecture.

Risks: *Hmm, that means this will probably happen to you.*

The consequences of one or more findings will be described as one or more risks. Risks are characterized in two ways:

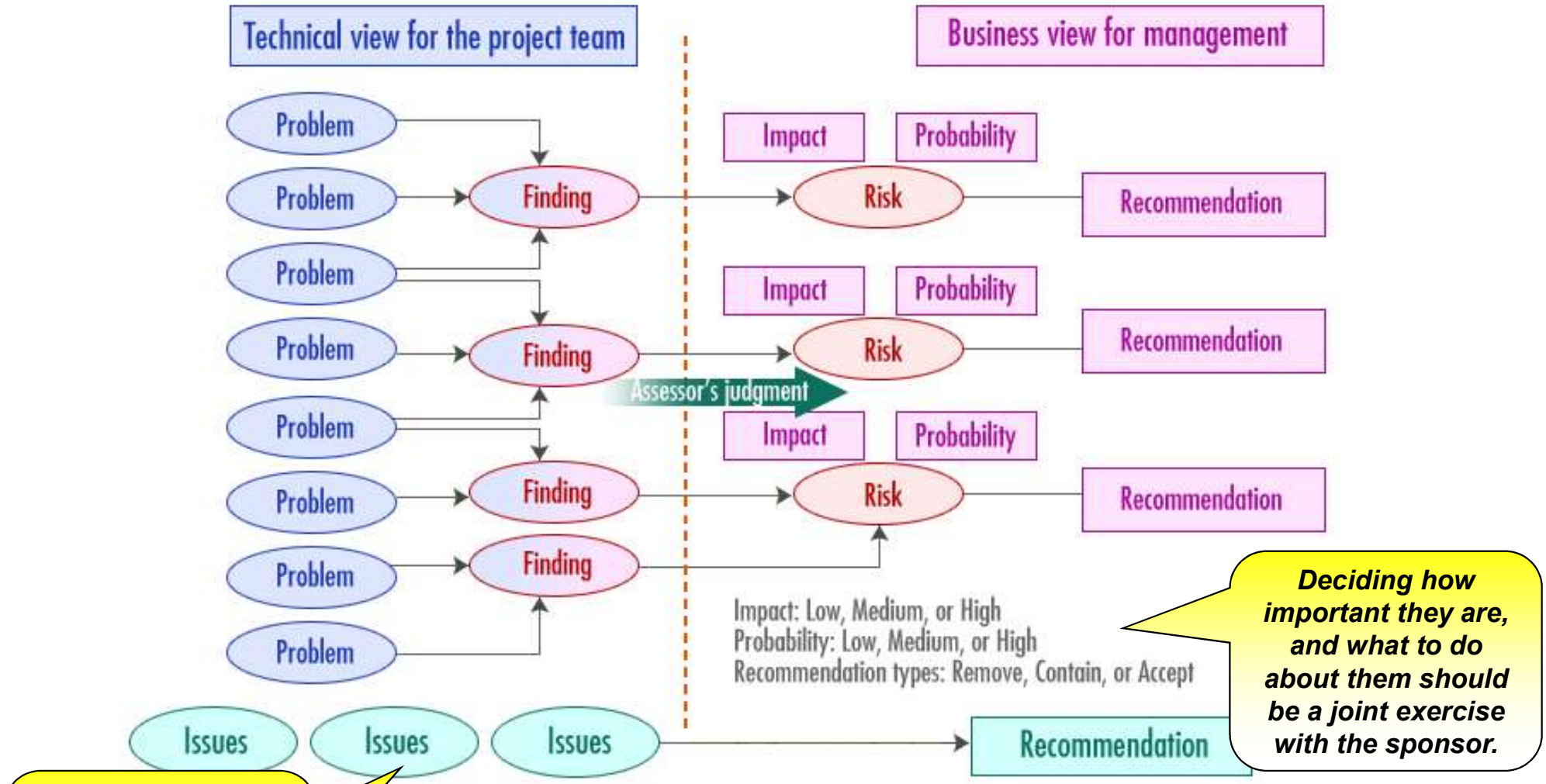
- Probability: How likely is it to happen?
- Impact: How bad will it be if it does happen?

Recommendations: *But if we do this, then maybe it won't!*

- If a risk is low probability and low impact, then it is likely to be simply accepted.
- But if it is high probability and/or high impact, then maybe an active decision has to be made about what to do.

Risk management is the process used to contain or mitigate known risks.

Assessments document problems, findings, risks and recommendations



Deciding how important they are, and what to do about them should be a joint exercise with the sponsor.

Establishing the symptoms and causes is a project task.

The problems, issues, and findings should be agreed to.

After risks are identified and a decision is made to do something about it, then what?

If a risk is high impact and/or high probability, then usually an active decision has to be made on what to do:

- **Accept** - “Doing nothing" is a perfectly acceptable option as long as everyone (sponsor and project team) agrees that it is OK.

- **Remove** – Alter something so that the risk goes away:
  - Change the requirements
  - Change the architecture
  - Change a constraint

- **Contain** – Establish a predetermined reaction that happens when the risk occurs and mitigates
its impact:
  - Additional development resource, time, or budget
  - Additional aspects of the design or deployed system
  - Human intervention, compensation, and so on

Assessments should also address assumptions, issues, risks and dependencies

RAID: Risks, assumptions, issues, and dependencies

- **Risk**: A possibility of something (usually negative) happening within the duration of a project
- **Assumption**: Things that the Solution Architect has taken on faith or otherwise have no direct reason to be thought of as true. Often, assumptions are implicit but are not recognized as such
  - Assumptions have to be verified and then become requirements or constraints. If found to be incorrect, the solution needs to be re-assessed.
- **Issue**: Something that is causing or will cause the solution to deviate from the requirements baseline
  - Issues have to be addressed, and recommendations for their resolution have to be established, agreed, and acted upon.
- **Dependency**: Something that a project expects to be in place or resolved before the project completes. Dependencies may be:
  - Business-centric: “The market will be ready for this project.” “The legislator will clarify that this approach to data sharing is legal.”
  - IT-centric: “The vendor will fix this bug.” “We’ll find a way of synchronizing these databases.”
  - Solution-centric: “The security server has the capacity to take our extra load.” “The warehouse management system will be ready in time.”

Module outline

- Introduction and objectives
- Viability
- Validation
- Approaches to viability and validation
- **Summary and references**

Summary

- A “good” architecture must be valid – i.e. solution to satisfy the requirements, and be viable – can be delivered and will work.
- Architectural assessment includes asking questions to challenge the architecture and associated delivery approach. A working architectural process delivers the base for a viable solution and its assessment.
- Validity and viability must be assessed throughout the project, as well as before (presales), and after (operations).
- Assessments can follow a well-known approach: Issues, Risks, Dependencies and assumptions:
  - Issues need to be resolved or architecture needs to be re-assessed.
  - Risks that are of high probability and/or high impact must be removed, contained, or accepted.
  - Dependencies need to be documented and managed during solution delivery.

References for viability and risk assessment

Risk assessment:

Managing Risk: Methods for Software Development, Hall E., Addison-Wesley, 1998, ISBN 0-201-25592-8

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [OMG-SPEM06] *www.ibm.com/legal/copytrade.shtml.*

Other company, product, or service names may be trademarks or service marks of others.

THANK YOU