IT Architecture

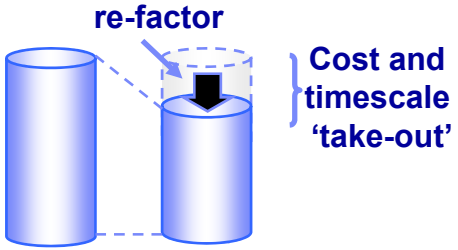
Solution Optimization -- An Engineering Approach

Dr. Marcel Schlatter,
IBM Distinguished Engineer
Member of the IBM Academy of Technology
marcel.schlatter@ch.ibm.com

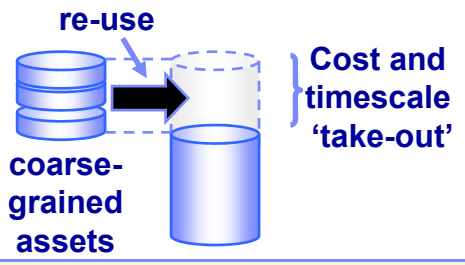
Solution optimization is an opportunity to match assets to your solution or requirements, with the help of an external team.

Finding the best solution for the business by:

1. Designing to price and simplifying the solution (e.g. by architecture refactoring)



2. Introducing assets to replace parts of the solution



Workshop is a formal checkpoint for re-use.

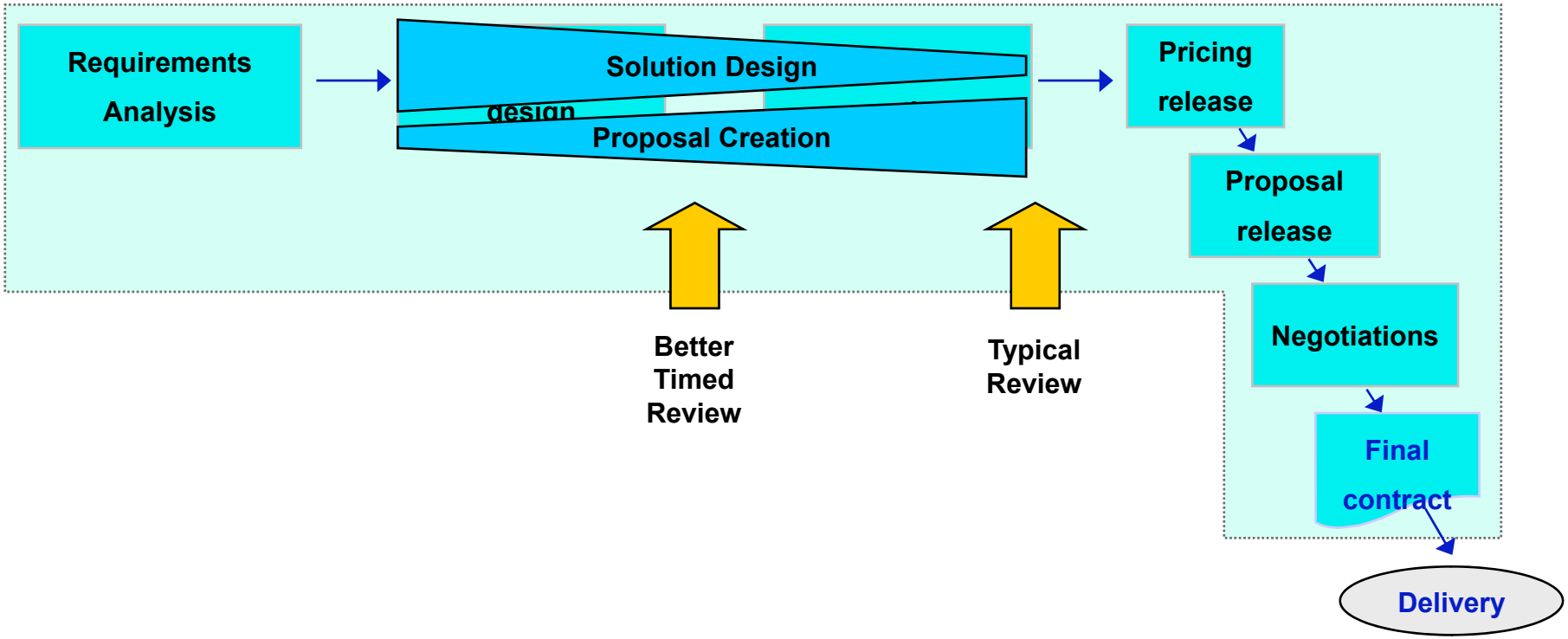
A natural place is just after the initial solution design.

From Solutions that Work to Solutions that Win

- Solution optimization's overall goal is to:
  - Maximise the attractiveness of the solution
  - Minimise the cost (and help to reduce price)
  - Win - Win
- This is achieved by applying optimization Patterns, including the re-use of assets, to optimise the solution design
- Bring in external people early in the engagement cycle to help shape better solutions
- Typical cost savings of 15%

Solution Design Process

Theory compared to “in practice”



Costs are often not considered early enough in the design process

- Solution designers often do not have an accurate understanding of the cumulative “costs” until late in the proposal process
- Available solution building blocks often do not come with an accurate estimate of costs that include:
  - Hardware
  - Prerequisite software (initial & ongoing)
  - Estimate of configuration (standard package)
  - Estimate of implementation time

Workshop Attendees

- Bid/Design Team
  - Bid lead with a deep understanding of the customer requirement and budget
  - Client IT Architect (if applicable)
  - Technical Solution Manager (Owner of cost model)
  - Lead Architect responsible for the end-to-end design of the solution
- Solution Optimization Team
  - Leader and team members with industry knowledge (e.g. Banking, Insurance, Telecom, Pharma, ...) and design/delivery experience with similar solutions to similar requirements

Workshop Inputs

- Customer requirements, e.g. RFP, strategy documents, etc.
- Customer budget or winning price
- Cost model
- Bid team view of major risks, issues and dependencies
- Solution overview, including system context diagram, architecture overview diagram, component model, operational model and deployment model

Osborn's Checklist [1]

1.Other uses?	New ways to use as is? Other uses if modified?
2.Adapt?	What else is like this? What other idea does this suggest? Does past offer parallel? What could I copy? Whom could I emulate?
3.Modify?	New twist? Change meaning, color, motion, odor, taste, form, shape? Other changes?
4.Magnify?	What to add? More time? Greater frequency? Stronger? Higher? Larger? Longer? Thicker? Heavier? Extra value? Plus ingredient? Duplicate? Multiply? Exaggerate?
5.Minify?	What to subtract? Smaller? Condensed? Miniature? Lower? Shorter? Narrower? Lighter? Omit? Streamline? Split up? Understate? Less frequent?
6.Substitute?	Who else instead? What else instead? Other ingredient? Other material? Other process? Other power? Other place? Other approach? Other tone of voice? Other time?
7.Rearrange?	Interchange components? Other pattern? Other layout? Other sequence? Transpose cause and effect? Change place? Change schedule? Earlier? Later?
8.Reverse?	Transpose positive and negative? How about opposites? Turn it backward, upside down, inside out? Reverse roles? Change shoes? Turn tables? Turn other cheek?
9.Combine?	How about a blend, an alloy, an assortment, an ensemble? Combine units?

[1] Osborn, Alex F. Applied imagination : principles and procedures of creative problem-solving. NY Scribner, 1957

Patterns (1 of 2)

- Simplify the Architecture
  - Is the solution over-engineered?
  - Has architectural elegance been included where it isn't really needed or adds little value?
  - Simplify complex architectures and excessive infrastructure layers.
  - Are we planning to build a Portal when all that is actually required is a handful of static web pages?
  - Are we including a database management system when the data model is so simple that a flat file or small collection of files would do?
- Integrate the Facilities
  - During the deployment of IT solutions, especially via an extended evolution process, it is likely that multiple services will have been provided in silos.
  - This can only be avoided by having an overall architecture on how all services will be provisioned together.
  - The technology may be consolidated to make best use of the available hardware, sharing to provide services, and reducing the duplication of functionality - for example authentication can be provided once across all service areas.
  - This will lead to a cost reduction associated with the provision of all services integrated together.
- Apply Frameworks or Skeletons
  - Reference architectures (Data Warehouse, Identity & Privacy, Open Source Middleware, Service Orientation, ...)
  - Apply a framework or a defined rigid development process with work-product templates or a skeleton program approach.
  - JSF (Java Server Faces)
  - MVC (Model – View – Controller)
  - SOMA (Service Oriented Modeling and Architecture)

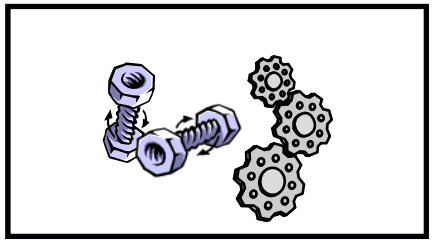
Patterns (2 of 2)

- Rightsize platforms and processors
  - Challenge accepted dogma around platform selection.
  - Could the solution be delivered equally well but at lower cost on a different platform
- Breakthrough the Application Design
  - Osborne's checklist can be a useful way to generate questions to challenge the traditional way of looking at things
  - Consider parallel vs serial processing, eg for database loads to reduce overall capacity or time requirements
- Optimize the AD environment
  - Typically there will be a variety of 80:20 rules at play; e.g. 20% of the application processes 80% of the workload or 80% of the critical functionality is supported by 20% of the system infrastructure.
  - Significant focus needs to be given to the critical 20%, but it may be possible to reduce cost by using, say, Rapid Application Development (RAD) techniques or other low-cost development approaches for the remaining 80%.

Utilize existing assets including architecture components, tools, processes, components and others

Tools

Components



Utilize assets where effective

Guidebooks

Other Assets

Make sure to use modern assets (not obsolete ones)

Based on practical sample cases

Use modern assets and utilize them where effective.
Obsolete assets increase the efforts in many cases.

Asset Granularity