Effective management of component-based system projects.

Author: Richard Veryard
Version: November 23rd 1998

richard@veryard.com
http://www.veryard.com

For more information about SCIPIO, please contact the SCIPIO Consortium.

info@scipio.org
http://www.scipio.org

Preface

Purpose of document

- To understand the impact of CBD on systems development projects.
- To understand the SCIPIO task structure.
- To understand how waterfall and spiral models map onto the SCIPIO task structure.
- To appreciate how SCIPIO supports a variety of project scenarios.

Acknowledgements

Introduction

A solution development process needs to be fast, flexible, simple, scalable and solution-oriented.

Fast	➢ Efficient execution of tasks ➢ Efficient handover / flow between tasks
Flexible	➢ Multiple entry points ➢ Multiple routes ➢ Context-sensitive
Simple	➢ Easy to understand - based on familiar metaphor
Scalable	➢ Suitable for large and small projects
Solution-Oriented	➢ Focused on delivery of solution

Software development lifecycles

Waterfall lifecycle

Many software development lifecycles adopt the metaphor of the waterfall.

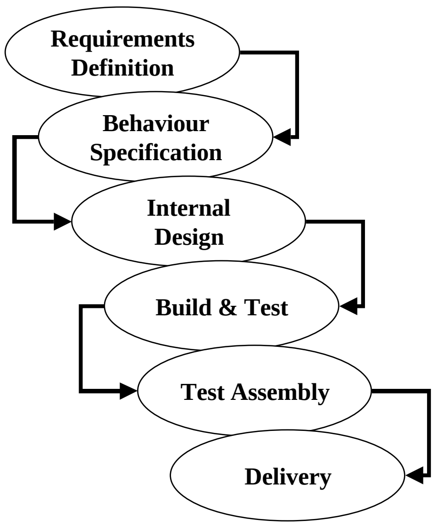


Figure 1: Typical waterfall lifecycle.

The word 'waterfall' is supposed to bring to mind a series of steps, with water flowing calmly and neatly from the top to the bottom. A contrived and controlled waterfall such as one might find in a public garden. Try not to think of the Niagara Falls.

Q What are the advantages of the waterfall approach? What are its drawbacks and limitations?

Q Have you been involved in any system development projects using the waterfall approach? Describe the experience. How well have these projects been controlled?

The waterfall approach attracts the scorn of a lot of software engineering experts, and it is often dismissed as obsolescent. However, despite its limitations, many large software projects continue to use a waterfall or modified waterfall approach.

Q Why do many software organizations and projects continue to use a waterfall lifecycle?

Furthermore, some proprietary CBD methods use the waterfall lifecycle, including Sterling Software's CBD method.

The waterfall approach is usually highly structured, with a fixed sequence of activities. As can be seen from Figure 2, it typically combines a *plan-do-test* loop with a top-down path from business concerns via application concerns to technology concerns.

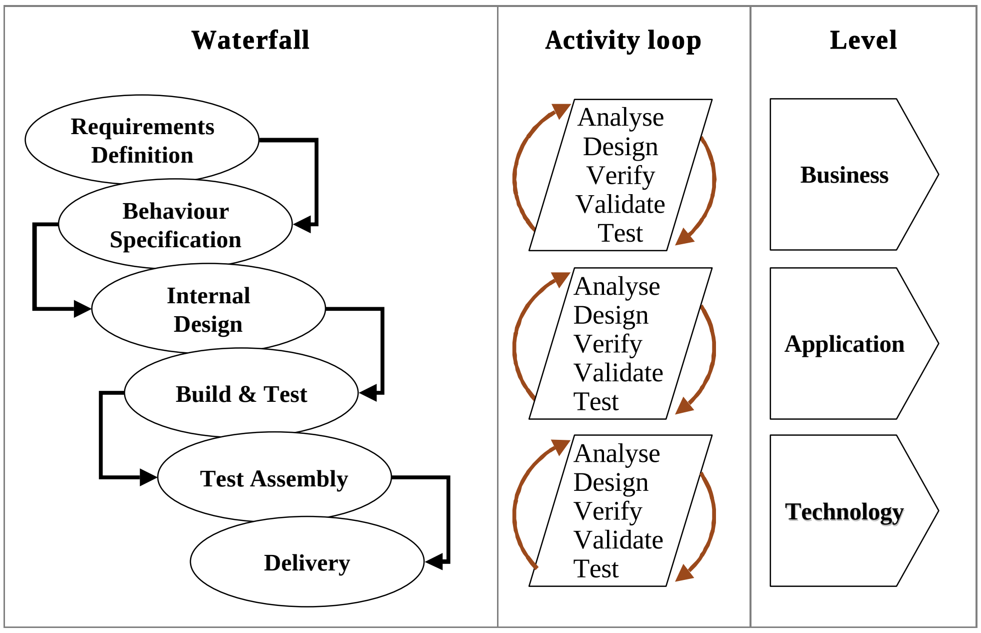


Figure 2: Waterfall separates activities and levels - but in a fixed sequence.

Iterative lifecycle

For a programmer, iteration means repeating something until something happens. For a systems developer, iteration means indefinitely refining and evolving and extending something. One popular version of the iterative approach takes the form of a spiral.

The iterative approach is supported by DSDM. Many CBD methods plump for an iterative approach, including the Select Perspective.

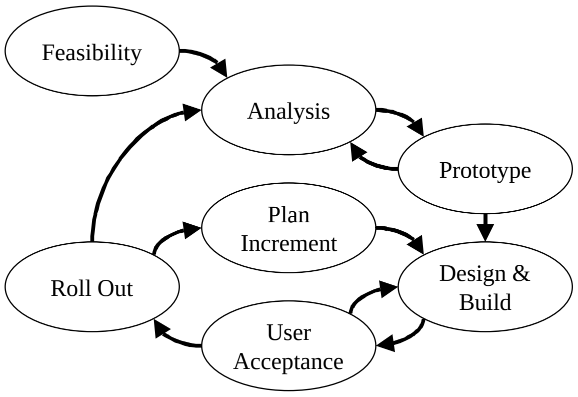


Figure 3: Typical iterative lifecycle.

Q What are the advantages of the iterative approach? What are its drawbacks and limitations?

Q Have you been involved in any system development projects using the iterative approach? Describe the experience. How well have these projects been controlled?

Each lifecycle has its advantages and disadvantages.

Q When would you use a waterfall approach? When would you use an iterative approach?

Q In what kinds of organization would you expect to find the waterfall approach used? In what kinds of organization would you expect to find the iterative approach used?

SCIPIO transcends the limitations of each lifecycle.

SCIPIO transcends the limitations of each lifecycle, including the waterfall and iterative lifecycles mentioned earlier in this section. It does this by defining the activities as collaborating parallel processes, independent of their sequence.

Each project may then impose a sequence on these activities, according to the particular requirements and circumstances of the project. SCIPIO defines a number of template plans, or **scenarios**, each of which traces a useful sequential path through the parallel activity structure of SCIPIO.

Thus although SCIPIO as a whole does not prescribe the sequence of activities, and is neutral in the choice of lifecycle, a particular SCIPIO scenario does prescribe the sequence of activities. A SCIPIO project may therefore adopt a waterfall scenario, or an iterative scenario, or a scenario based on some other lifecycle.

Towards a service-based approach to project management

SCIPIO applies component-based thinking to business organizations and software architectures. We also find it useful to apply the same component-based thinking to the structure of a project.

Traditional projects have been run as relay races.

Traditional projects have been run as relay races, with each activity passing an increasingly heavy baton (the so-called deliverables) onto the next activity. (This metaphor applies equally to waterfall and spiral development.)

With the traditional approach, information and detail accumulates exponentially during a project or series of projects. This can cause problems even for small projects and systems.

Above a certain size, the sheer volume of project information can severely handicap productivity and quality. This is one of the many reasons why small projects are easier to manage, and have a higher success rate, than large projects.

Rapid delivery demands parallel activity

Fast delivery of large complex solutions requires maximum parallelism.

As major manufacturers have found, the traditional sequential approaches to product development cannot deliver products to market fast enough. Instead, all phases of product design / engineering and production design / engineering must be carried out concurrently.

This parallelism even applies to the architectures, which are elaborated and refined as the detailed design and construction work progresses. (An Aer Lingus building near Dublin Airport was recently completed according to concurrent engineering principles.)

This of course places new demands on project management. Schedules will use end-to-end dependencies rather than the (simpler) end-to-start dependencies. Team structures will be more complex, with access to a far greater range of skills required than one person can master. Coordination must be carefully balanced - neither too little nor too much.

The days of the generalist are numbered.

One of the ideas behind Information Engineering was the idea of an empowered software developer, able to span the system lifecycle from Business Analysis through to Software Construction and Testing. This idea was supported by two key facts:

- The same notations served both business analysis and software design.

- Many of the technical skills of the programmer were now redundant, as they had been supplanted by software tools (such as CASE tools).

Many organizations took this idea very seriously, and were reasonably successful in defining a generalist job of system developer. However, some organizations failed (for a variety of reasons), or chose to stick with a traditional separation between business-oriented systems analysts and technology-oriented software builders.

But technological development makes information systems development dependent on an increasing array of specialist areas. The sheer variety of specialist skills demanded of systems development makes it increasingly unlikely that system developers or system architects will be able to master the growing range of specialist areas.

Q How many specialist areas can you name, that might be relevant to information system development projects in your organization?

The days of the generalist are numbered. Among other things, this means that we should abandon the expectation that a single individual can comprehend and assimilate all the documentation produced by the project.

Each area needs specialist knowledge and methods.

Each specialist area needs its own specialist knowledge and methods. It collects and analyses information that may not be meaningful elsewhere.

Physical database design is a good example of this. Database specialists have private techniques and notations that are not visible to anyone else.

Complexity should be hidden behind clearly defined service interfaces.

To make projects more manageable, the complexity of information and techniques should be contained within clearly defined modules of activity, each of which provides a defined service to the project manager.

For example, consider the activity of locating and evaluating externally-sourced components. Early in a project, the project manager may ask a member of the project team to conduct a preliminary survey in a given functional area, to confirm that components are available. The person may report back that there are at least six candidate components, ranging in price from zero dollars (shareware) to 200 dollars. Much later in the project, the project manager asks the same person to review the candidates and recommend one. This can be regarded as a continuation of the first activity.

Note that we could schedule the activity to be performed at one time, but this may not be the best option. It usually makes sense to carry out the initial search as soon as possible, so that the project manager knows which parts of the requirement can be satisfied by bought-in components and which parts will need to be developed in-house. And it usually makes sense to carry out the actual selection as late as possible, because new components are being published all the time.

In traditional waterfall approaches, the component selection short-list would become part of the common project knowledge base. As we have seen, projects accumulate large quantities of information of this kind. A small trickle of information at the start of a project becomes a huge flood towards the end.

By encapsulating such information as the component selection short-list within a particular project activity, this gets away from this information flood.

Not only the information but also the specific techniques may be hidden within a project activity. The project manager may not always need to know how the person conducts the search, nor what specific search and selection criteria are used, and may merely need to set some broad constraints on the activity.

For another example, physical database design is often provided as an external service to the project, rather than carried out by members of the project team.

Q In a traditional database development project, what are the points of contact between the database group and the rest of the project team?

Projects can be designed as collaborating activities.

We have already seen that both business processes and software applications can be designed as collaborating activities. We now see that the same is true for projects.

Systems development involves a number of interacting activities, which provide services to each another. These activities may be carried out in sequence or in parallel. Some of these may be under the direct control of a project manager, while others are provided by separate teams.

Centralized project management is clearly one way of organizing all these project activities. Various forms of twin-track development are also possible, and may be recommended for many organizations. However, a general process framework simply needs to identify the services at a high level.

Some project activities are resource-intensive while others operate in the background.

Project activities may operate in two modes: intense and background. When operating in intense mode, a project activity consumes significant resources; when operating in background mode, a project activity has low resource requirements but may benefit from access to information and administration services.

A project activity may switch between the intense mode and the background mode. For example, a project manager may request an investigation of specific business opportunities with the Internet. This is a time-boxed activity, which may be managed as a subproject, resulting in information or ideas leading to a proposal. When this is achieved, the subproject ends.

However, the thought processes continue in background mode. The people engaged in the original investigation may have further ideas, or may discover additional information. If the new ideas or information are sufficiently exciting or challenging, the people may ask management to sponsor another investigation, thus returning the same project activity to the intense mode, perhaps for another couple of weeks, before dropping back into background mode again.

Some project activities operate exclusively in background mode. These are essentially monitoring activities, requiring little or no resource, capable of detecting events and triggering other activities. They do not usually appear on project schedules, but they do appear on job descriptions, since key project staff may have the responsibility and authority to trigger other activities.

Background activities may sometimes be essential, but they are often under-appreciated by management, and are performed by conscientious staff as 'skunk works'.

SCIPIO defines all project activities, including background ones. This helps ensure that skills and infrastructure are in place to support all project activities.

We use object modelling techniques to define the collaborations between project activities.

One of the beauties of object modelling, of course, is that it allows us to show interactions (or collaborations) between several activities at an abstract level, without specifying method, sequence, or management boundaries. We can then make these decisions explicitly for a given organization or project scenario.

Q Which diagram shows the interactions between objects without specifying their sequence? Which diagram shows the sequence of interactions?

Project activities can be grouped together for management purposes.

There is much talk of "twin-track" component-based development. What this means is that a management boundary has been drawn between two sets of activities. Typically, we find those concerned with building components on one side of the boundary, and those concerned with using components on the other side.

Another form of "twin-track" development can be found in the renewal of legacy systems. One track includes those activities directly associated with delivering new functionality or new access channels to the business, while the other track includes those activities concerned with restructuring the legacy systems architecture, harvesting components and identifying opportunities for making the system more flexible and robust.

In some cases, it may be appropriate to define three or more tracks.

Often the primary reason for separating the tracks is a difference of timescale. One track may have very short timescales, responding to urgent business demands within a few weeks. Another track may have longer timescales, performing extensive development or restructuring work according to a strategic programme.

There are various other reasons why we might wish to divide development activities into separately managed tracks. The division may be based on existing organizational boundaries, or existing teams with particular capabilities. Alternatively, they may be based on a logical analysis of the interactions between the activities, with the tracks defined to increase cohesion and reduce coupling. Often the division will be a pragmatic one, modifying existing structures to suit the logical requirements of the whole development programme.

The boundaries may represent interfaces between organizations. Such interfaces may be defined as contracts. There may be benefits in adopting standard industry templates for these contractual agreements, but this will not always be the best option.

The management approach may extend or modify the project activities.

Many methods call for iteration of tasks. A given design technique may be executed at one stage of the project to produce a ‘first cut’ deliverable, and then may be re-executed at a later stage of the project to produce a ‘final’ deliverable.

We regard the control of such iteration as a project management concern. For this reason, we do not explicitly specify iteration and related matters within the solution development task lists. Instead, we leave them to be ‘plugged in’ as part of the project management approach. In other words, the task iterations are to be generated when the project plan is created, based on the preferred project management approach.

In general, we recommend a RAD approach to iteration. The DSDM consortium, which has developed some industry-wide recommendations on RAD, suggests that, to control iteration, each important design technique be iterated three times. However, some situations or organizations may prefer to plug in an alternative approach.

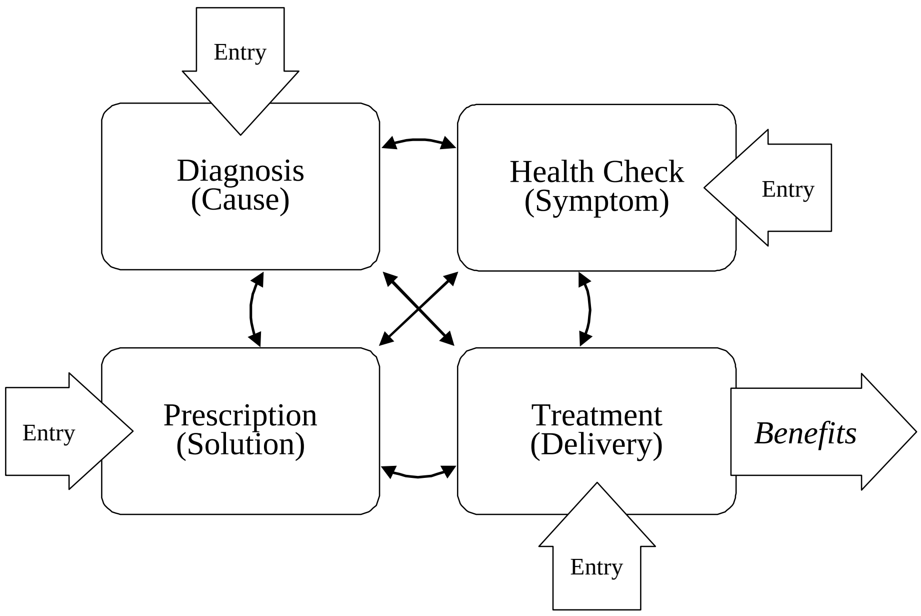
Furthermore, the actual project management tasks, such as project planning, are assumed to be part of the project management plug-in, and are not contained within the Solution Provisioning task lists.

Similarly, the quality management tasks, such as quality planning, training, reviews and audits, are not contained in the Solution Provisioning task lists but are plugged in as part of the preferred Quality Management approach. The SCIPIO consortium is able to advise on a standard Quality Process, based on ISO 9000, TickIT and the SEI Capability Maturity Model, but SCIPIO is open to alternative quality management approaches being plugged in.

SCIPIO activity structure

Solution Provisioning

Our structure is based on a medical metaphor.



Imagine that the medical profession operated according to the best principles of software engineering. If you had a headache, you'd need to start by having a full brain scan. Some weeks or months later, the analysed results would be returned from the laboratory to your doctor, who would study them carefully to determine the exact cause of your headache. Based on this analysis, the doctor would formulate a complex solution, which might involve a combination of things: a change in your diet perhaps, more frequent exercise, minor surgery or physiotherapy. Then, and only then, would you be permitted to take a headache pill.

Most people start by taking a headache pill. The doctor only gets involved if the pill doesn't have the desired effect, or the headache doesn't go away. The full brain scan is kept as a last resort. Furthermore, even when the doctor agrees that a brain scan is necessary, the sufferer may be allowed to continue taking the headache pills.

Thus the medical headache-solving cycle can start at any point. And the stages can be carried out in reverse order, or concurrently, depending on the situation.

In business problem-solving and information systems development, the traditional approach attempts to do all the analysis at the beginning of a project, and all the delivery at the end.

Q What are the consequences of this approach? How successful is it?

The SCIPIO activity structure has the same four basic components, although we have relabelled them to make them more relevant to business problem-solving and information system development.

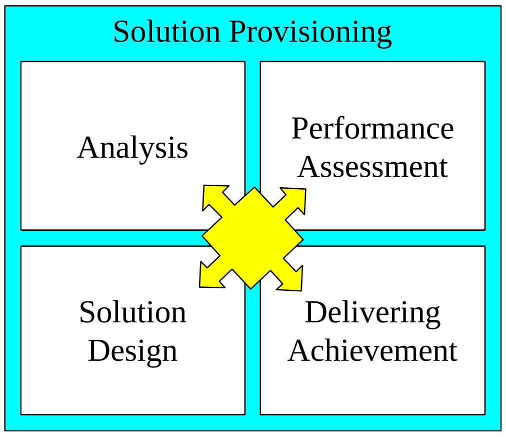


Figure 4: Solution Provisioning Structure

Performance Assessment

Performance Assessment (sometimes referred to as System Assessment) considers the characteristics of the system from several viewpoints, including:

- Business process performance (including cycle time, cost and quality of service);
- Application characteristics (typically the quality characteristics identified by ISO 9126: functionality, maintainability, usability, reliability, efficiency and portability);
- Technological infrastructure characteristics (including value-for money, technical performance and flexibility).

There are six types of performance assessment, which may be considered at different stages of the project. These are:

Evaluation: *An estimation or measurement of past or present performance.*	***Prediction:*** *An estimation or declaration of future performance.*
1. Current performance of system. This is typically what triggers the improvement project.	2. Target performance (or objective setting) of system.
3. Benchmark performance of system. Benchmarks may be internal or external.	4. Simulated performance of current system. This is an optional step; it provides a confirmation of our analysis, by showing that our model of the existing system is consistent with the observed performance and is explained by it.
6. Actual performance of redesigned system. This is an evaluation of the success of the project, and may result in further improvement projects.	5. Simulated performance of redesigned system. This provides a prediction of the business benefits of a proposed solution, and an identification of any associated risks. This provides a confirmation that the solution is likely to work.

Thus many of the assessment subtasks can only be carried out after some analysis, design and/or delivery subtasks have been carried out.

Analysis

The purpose of Analysis (sometimes known as Diagnosis) is to understand the structure of what already exists, and to identify the causes of any problems or restrictions. A series of models of the current situation is produced, showing how various people, roles, departments, companies and systems interact. These models (known as As-Is models) serve several purposes:

- They provide an explanation of the current level of performance, both in absolute terms and in comparison with benchmarks. They allow the immediate symptoms of performance problems to be traced to their underlying causes.

- They provide a basis for the design of an improved business process, and provide a basis for the possible restructuring and redeployment of parts of the current system to support the redesigned business process.

Design

Solution Design includes the **definition** of the solution as well as its **planning and sourcing**. The improved (To-Be) business process is represented in the same way as the current (As-Is) business process: as a set of interactions between people, roles, departments, companies and systems.

- The interactions may be improved and made more effective.

- Wholly new components may be introduced into the system. These may be business components, application components or technological components.

- Existing components may be adapted to fit the improved system context in which they are to operate. Opportunities are identified to make the existing components and their interfaces more generic, to enhance future adaptability.

- The business rules are reformed and mapped onto the components. Each component is assigned the responsibility for maintaining one or more rules.

- A holistic picture of the solution is produced.

- An implementation plan is developed, indicating the selected source for each new or modified component.

Delivering Achievement

Delivery divides into **Component Acquisition, Component Development**, **Assembling** and **Commissioning**. The provisioning of the new components may follow a variety of paths, including:

- Existing components may be wrapped, repackaged or restructured.

- Modifications to existing components may be hand-crafted. Alternatively, if a suitable component already exists elsewhere, it may be cheaper and easier to replace the old component with an improved version satisfying the requirements.

- New components may be bought ‘off-the-shelf’ from a component library. Or a component with similar description may be bought and customized.
- Finally, and only where necessary, new components may be commissioned from component builders, either in-house or external.

These considerations apply to the provisioning not only of software application components but also of other pieces of the solution, including business process components (such as user guides and stationery) and infrastructure components (such as hardware and system software).

The development and maintenance of components will normally follow a prototyping approach. Furthermore, whole business solutions may be assembled and piloted before roll-out to the remainder of the target organization - this can be regarded as a business-level equivalent of a prototyping approach.

Note: the planning of development and implementation is regarded as part of project management.

Solution Management

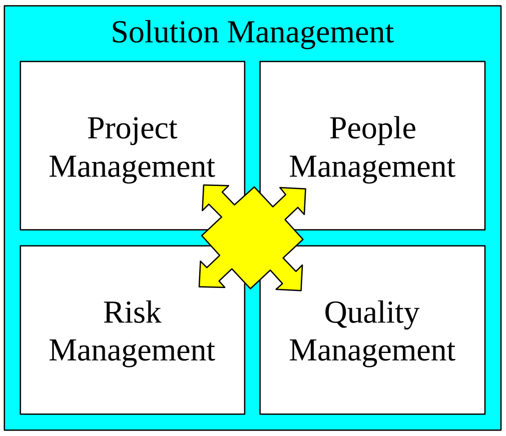


Figure 5: Solution Management Structure

In this document, we do not provide a detailed account of the solution management activities. We assume that any reasonable management approach can be plugged into a SCIPIO project.

Summary

If we put the Solution Provisioning activities together with the Solution Management activities, we get the full SCIPIO activity structure as shown in Figure 6.

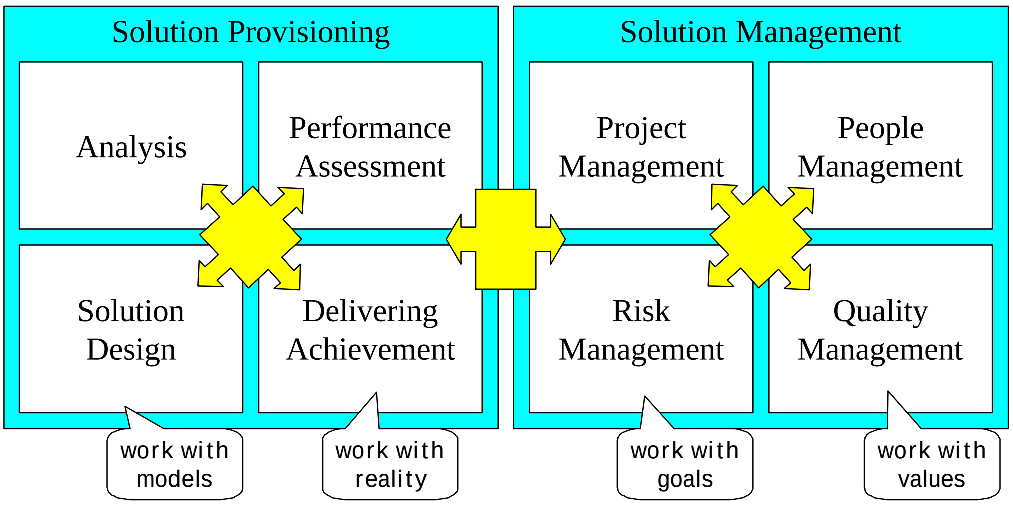


Figure 6: Full activity structure of SCIPIO

Note that this structure can be seen in four sections:

- In Analysis and Solution Design, we are primarily working with **models** (As Is and To-Be respectively).
- In Performance Assessment and Delivering Achievement, we are primarily concerned with '**reality**' - with the characteristics of the 'real' business process or system.
- In Project Management and Risk Management, we are primarily concerned with project **goals** and the factors that lead to their success or failure.
- In Quality Management and People Management, we are primarily concerned with the personal, collective and corporate **values** that are realised through project work.

Projects and Scenarios

SCIPIO covers a broad range of project scenarios from application assembly to BPR. In this chapter, we identify four main solution-oriented scenarios, each with some variations.

- Business Process Improvement
- Reengineering
- Object Development from Scratch
- Middleware

SCIPIO identifies three different flavours of improvement, at each of the three levels (business process, application and technology). This is based on the wisdom that it usually makes sense (at all three levels) to simplify before integrating ("Don't pave the cow paths!"), and to integrate before transforming. The four scenarios cover all nine variations, as shown in Figure 7.

Business Process	**Simplification** Using IT to eliminate redundant steps in a business process, or to reduce excessive variety.	**Integration** Using workflow management tools and new software components to link business processes together more effectively.	**Transformation** Radical BPR
		Scenario: Business Process Improvement	
Applica-tion	**Simplification** Using CBD in a tightly focused way to eliminate specific problems in legacy systems.	**Integration** Using middleware to link information systems together more effectively.	**Transformation** Building new Information System solutions using commercially available components.
	Scenario: Reengineering		*Scenario: Object Development from Scratch*
Infra-structure	**Simplification** Replacing unnecessary complications or variations in the technological infrastucture.	**Integration** Linking heterogeneous technologies together more effectively.	**Transformation** Satisfying new technical requirements, or exploiting new technological opportunities.
	Scenario: Middleware		*Scenario: Object Development from Scratch*

Figure 7: Four Project Scenarios, covering nine kinds of improvement

References

Dynamic Systems Development Method (DSDM) is managed by the DSDM Consortium. See their website at http://www.dsdm.org/

SCIPIO. For more information on SCIPIO, including a detailed task structure, please see the SCIPIO website at http://www.scipio.org/

Select Perspective is described in Stuart Frost & Paul Allen, Component-Based Development for Enterprise Systems: Applying the Select Perspective. SIGS Books and Cambridge University Press, 1998. See also the Select Software Tools website at http://www.selectst.com/

Sterling Software CBD Method. This is embedded in Sterling's Advisor product. See http://www.cool.sterling.com/cbd/