Aims, Principles and Structure

Editor:	Richard Veryard
For:	SCIPIO Consortium
Status:	Beta Version
Version:	0.9
Filename:	SCIPIOap.pdf

Contents

Introduction

Purpose of this document

This document describes the aims, principles and structure of SCIPIO. It is intended for three groups of reader:

- Business and IT management, wishing to understand the opportunities of SCIPIO.
- Method practitioners and experts, wishing to evaluate SCIPIO.
- Tool managers, wishing to understand the implications of SCIPIO for tool support.

Structure of this document

Most descriptions of a methodology are one-sided. They focus on a task structure, or a set of notations, or a set of design principles, or the use of a particular tool.

To counteract this tendency, and to give you an all-round description of the SCIPIO methodology, there are five things we want you to know about.

It turns out that these five things correspond to the five 'elements' of ancient Chinese wisdom: fire, earth, metal, water and wood. So we shall use this metaphor as a memory-jogger.

Purpose and motivation	What benefits does SCIPIO offer, to whom, under what circumstances?	*Fire*
Insights and new thinking	What breakthroughs (or "paradigm shifts") does SCIPIO offer?	*Water*
Conceptual architecture	How do the ideas of SCIPIO fit together?	*Metal*
Task structure	How are the ideas of SCIPIO translated into action?	*Wood*
Theoretical and practical foundation	How does SCIPIO build on established knowledge and experience?	*Earth*

This document provides a brief description of each element of SCIPIO.

Acknowledgements

Valuable contributions to this document have been made by other members of the SCIPIO Consortium.

Summary of SCIPIO Principles

- SCIPIO designs business processes to support business relationships and interactions (e.g. with customers) rather than to support existing task structures and hierarchies.
- SCIPIO uses consistent concepts and principles for business design and technological design.
- SCIPIO creates software components by wrapping legacy code, wherever possible.
- SCIPIO designs applications as a series of software components, linked together by flexible middleware, assembly tools or workflow engines.
- SCIPIO evolves and implements solutions organically, in small but coherent steps.

SCIPIO designs business processes to support business relationships and interactions.

WHY	HOW
➢ This makes the business processes more flexible, and more capable of learning. ➢ We don't just want to support existing task structures and hierarchies.	➢ We concentrate on modelling conversations, through the concept of **exchange**.

SCIPIO uses consistent concepts and principles for business design and technological design.

WHY	HOW
➢ We want to have simple and coherent models to discuss with all stakeholders. ➢ We want to be able to trace rigorously between business requirements and technological solutions.	➢ One set of modelling concepts and notations. ➢ Based on UML, with important additions.

SCIPIO designs applications as linked systems of independent software components.

WHY	HOW
➢ This means we can rapidly reconfigure a system in response to changing business demands.	➢ Components are obtained from the most cost-effective source, and operated on the most convenient technical platform. ➢ Components are linked together by flexible middleware, assembly tools or workflow engines.

SCIPIO creates software components by wrapping legacy code, wherever possible.

WHY	HOW
➢ This means we can greatly reduce the cost and risk of new applications. ➢ This means we can address the Year 2000 and EMU problems with component-based development.	➢ We define the vocabulary and behaviour of each identifiable subsystem. ➢ We freeze interfaces we don't want to reengineer. ➢ We reengineer interfaces we don't want to freeze. ➢ We build bridges between old subsystems and new components.

SCIPIO evolves and implements solutions organically, in small but coherent steps.

WHY	HOW
➢ We want to get a set of 'quick wins' for the business. ➢ We want to deliver software applications before the requirements change. ➢ We want to focus organizational change on one group of users at a time.	➢ Define interfaces at business level and at application level. ➢ Create local processes that connect into wider process. ➢ Manage application development as a series of small (RAD-like) projects within a larger coherent development programme.

Purpose: SCIPIO aims to improve the relationship between business and IT.

SCIPIO aims to achieve the following improvements in the relationship between business and IT. This represents the purpose or motivation of SCIPIO.

- Value-for-money for business from IT.
- Business process improvement through IT.
- IT stimulating business advantage.
- Investment protection for IT assets.
- IT enhances business capability and flexibility.
- Reduce the business risk of IT
  — and the IT risk of business.

SCIPIO aims to ensure value-for-money for business (from IT).

SOME PERSPECTIVES	SOME RISKS
CEO Are we spending enough on IT? **Finance** Are we spending too much on IT? **IT** Are we contributing as much as possible to the business? **Supplier** How can we cost-justify our products and services in terms of business benefit?	➢ IT systems not aligned to business needs. ➢ IT systems cost too much to build, operate and maintain. ➢ IT personnel unable to communicate effectively with business personnel. ➢ Business opportunities not discussed with IT until too late.

SCIPIO aims to enhance the contribution of IT to the business.

- To ensure value-for-money.
- To ensure that the level of expenditure is appropriate and in proportion
  - to the size and complexity of the business
  - to the potential benefits to the business.

SCIPIO aims to get business process improvement (with the aid of IT).

SOME PERSPECTIVES	SOME RISKS
CEO Can IT help us grow the company?	▪ Loss of business focus for IT.
Finance Can IT help us cut costs?	▪ BPR solutions too simplistic.
Sales Can IT make our systems more flexible for the customer?	▪ Business improvements delayed while IT systems are developed.
HR Can IT make our systems more flexible for the user?	▪ IT designs reduce opportunities for business process improvement.
	▪ Simple IT requirements escalated into major redevelopment programme.

SCIPIO aims to help IT stimulate business advantage.

SOME PERSPECTIVES	SOME RISKS
CEO Does IT offer us something new and exciting?	▪ Technical innovation for its own sake.
Supplier You should be the first company in your market to have this amazing new capability.	▪ Escalating costs of leading edge ("bleeding edge").

SCIPIO aims to protect the business investment in IT assets.

SOME PERSPECTIVES	SOME RISKS
Finance We've sunk millions into these systems, and they've only been running two years or so. Was that investment wasted?	▪ Creation of inflexible new systems with built-in obsolescence.
IT How can I cost-justify new IT systems, if the business isn't getting a demonstrable ROI?	▪ Inability to extract and reuse value embedded in existing systems.
Supplier How can I persuade them to buy more of my products and services?	

SCIPIO aims to enhance business capability and flexibility.

SOME PERSPECTIVES	SOME RISKS
CEO Does IT stop us changing the company?	▪ Business initiatives blocked or delayed because of IT constraints.
Finance Do changes in the company incur high IT costs?	▪ Business initiative incurs high cost for IT system alteration.
Sales Does IT prevent us doing what our customers want?	▪ Business unduly influenced by past IT decisions.
HR Does IT enhance or reduce the quality of life for our people?	▪ Business thinking restricted by existing patterns of data.

SCIPIO aims to reduce the business risk of IT — and the IT risk of business.

This section presents the risks identified in previous section. Thus there is an accumulation of risk. One of the aims of SCIPIO is to manage this risk.

➢ Business initiatives blocked or delayed because of IT constraints.	➢ Business initiative incurs high cost for IT system alteration.
➢ Business unduly influenced by past IT decisions.	➢ Business thinking restricted by existing patterns of data.
➢ BPR solutions too simplistic.	➢ Business improvements delayed while IT systems are developed.
➢ Creation of inflexible new systems with built-in obsolescence.	➢ Technical innovation for its own sake.
➢ IT systems not aligned to business needs.	➢ IT systems cost too much to build, operate and maintain.
➢ IT personnel unable to communicate effectively with business personnel.	➢ Business opportunities not discussed with IT until too late.
➢ Inability to extract and reuse value embedded in existing systems.	➢ Loss of business focus for IT.
➢ IT designs reduce opportunities for business process improvement.	➢ Simple IT requirements escalated into major redevelopment programme.
➢ Escalating costs of leading edge.	➢

Insights

Our assumptions are challenged by the new agenda of business and technology. New technologies (especially CBD and ODP) allow some of the old dilemmas and difficulties to be overcome - but create new dilemmas and difficulties in their place.

We need to move beyond traditional development approaches.

Beyond greenfield development

For most IT developments today, the greenfield development approach is no longer acceptable. Technology provides several new mechanisms for reusing portions of legacy systems.

Beyond waterfall development

Although we are often told that the waterfall approach is obsolete, large numbers of IT organizations still rely on this approach. For one thing, it appears to provide a level of manageability - especially on large or distributed projects - that is missing from some of the rival approaches. Where some of the phases of development are outsourced, the waterfall approach appears to offer the most reliable way of negotiating and managing software procurement.

However, there are serious and well-documented problems with the waterfall approach. Rival approaches appear to offer answers to these problems, but bring problems of their own.

We therefore need to find a way of combining the known advantages of the waterfall approach with the known advantages of a prototyping or evolutionary approach.

We need to hide complexity

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.

It makes sense to carry out the initial search as soon as possible, so that the project manager knows which parts of the requirement are likely to be satisfied by bought-in components and which parts will need to be developed in-house. But it 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 the project progresses, it accumulates 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.

Requirements change

In a traditional waterfall approach:

- the analysis is supposed to be complete before the design can start;
- the design is supposed to be complete before construction and unit testing can start;
- construction is supposed to be complete before system testing and acceptance testing can be carried out;
- and all testing should be complete, with resolution of all issues, before the system can be implemented.

Even when radical business process reengineering is undertaken, it will usually be impractical to stop all other initiatives until the business modelling is complete.

SCIPIO recognizes the practical inevitability of requirements change. Business and technology changes are monitored throughout a project. Relevant changes are filtered, analysed and fed into the project in a controlled way. This is carried out in parallel with planned development activities.

We need to balance a general approach with specialist skills.

A notation is not a technique. A technique is not a skill.

Although we often use the same notations for solving business problems and for solving technical problems, this does not mean that we equate business problems with

technical problems. Different techniques are required, and although there are some general patterns that apply across different areas, there are many patterns that are specific to one area. Skills and experience in one area do not automatically confer abilities in another area.

We cannot all be experts in everything.

As technology grows more complex and sophisticated, and is applied to an increasing breadth of business and administrative opportunities, the number of specialist areas continues to increase. Even on a small project, it is unlikely that one person will command all the knowledge and skills required.

Furthermore, no one person can reasonably assimilate the sheer quantity of information produced during a typical project.

If we apply component-based thinking to the development method itself, we can localize (or "encapsulate") specialist considerations.

In previous methods, this would typically be done with such specialist concerns as database design. Within a software development project, only a few people needed to be familiar with the internal design of a particular database platform; the rest of the project would use logical data models. (In effect, the logical data model served as the interface between the database designer and the rest of the project.)

We need to balance object-orientation with process-orientation.

It is often useful to consider processes as objects.

This is a central premise of the "object-oriented" movement. It is sometimes called "reification".

It is often useful to consider objects as processes.

Conversely, we need to look at the dynamic and managed aspects of things. Here are three examples.

Application as Thing		Application as Process
Fixed lump of software code.	→	Dynamically reconfigurable set of information and communication services.

Organization as Thing		**Organization as Process**
Fixed allocations of power and resources.	→	Growth of individuals and groups in capability and maturity.
Business Relationship as Thing		**Business Relationship as Process**
Fixed contract between supplier and customer.	→	Mutual responsiveness and partnership.

Open Distributed Component-Based Systems do not conform to traditional assumptions.

Beyond two-valued logic

In a closed non-distributed system, all enquiries have a definite answer. For example, either the customer record is found, or it is not found.

The behaviour of a component of such a system can therefore be specified using simple two-valued logic. If condition is TRUE then action or outcome is X; if condition is FALSE then action or outcome is Y.

In a distributed system, an enquiry may not have a definite answer. One simple reason for this may be that a remote server containing customer data is currently unavailable.

To specify the behaviour of a component of a distributed system, we may need to use three-valued logic. If condition is TRUE then action or outcome is X; if condition is FALSE then action or outcome is Y; and if condition is UNKNOWN then action or outcome is Z.

(A logically equivalent alternative to using three-valued logic explicitly is to define lots of exception conditions. But this tends to make the specifications more complicated.)

In an open distributed system, there are many other reasons why the results of an enquiry may be indeterminate. For example, an Internet search using the same search engine with the same parameters may not yield the same results twice.

Beyond traditional testing

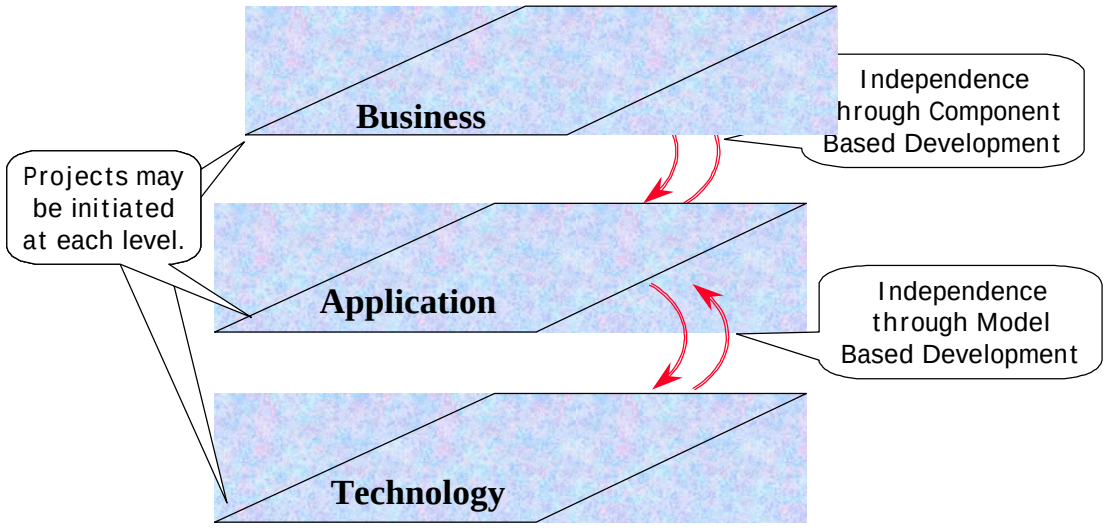
Besides making the specification of components more complex, the indeterminacy referred to in the previous section makes it much harder to test and debug systems and components, as it often proves impossible to replicate anomalous results.

Furthermore, ODP challenges the traditional double negative of testing: if a component fails to fail a test, then it must satisfy requirements.

Architecture

SCIPIO separates different viewpoints of a system or solution.

We can separate the application from the business and technology layers.

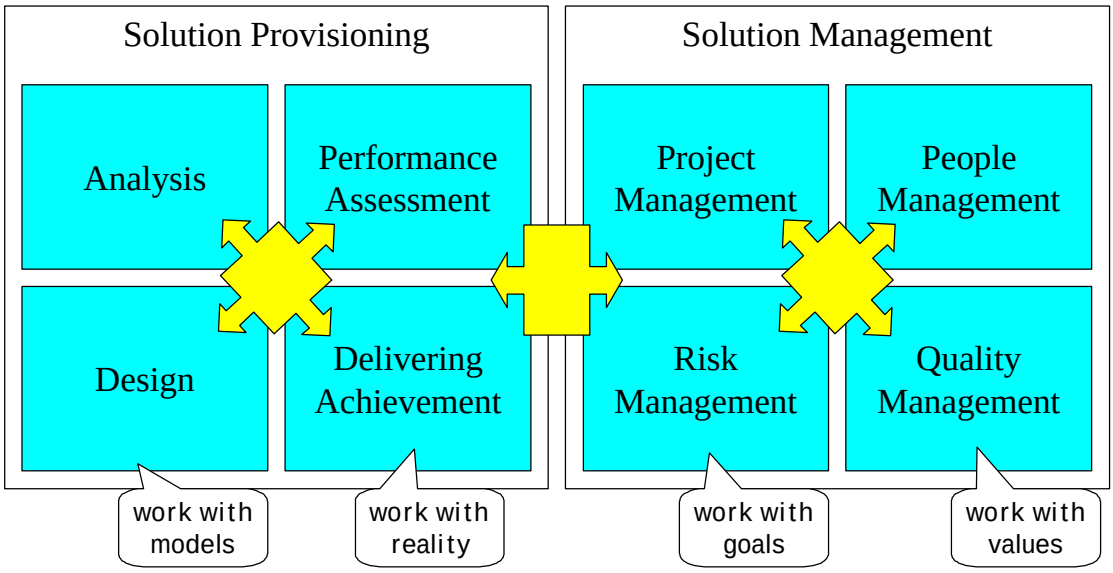


We can define five separate viewpoints.

Going further than the three layers identified in the previous section, the ODP Reference Model (ISO 10746) defines five separate viewpoints on an open distributed system. We generalize this as follows:

Enterprise (WHY)	*Process as Collaboration*	Set of **responsibilities** for maintaining invariants and achieving goals.
Transaction (WHEN)	*Process as Exchanges*	Set of **authorities** to participate in defined transactions.
Behaviour (WHAT)	*Process as Activities*	Set of **operations** with defined pre- and post-conditions.
Design (HOW)	*Process as Services*	Set of published **interfaces**, each offering a cluster of operations.
Physical (WHERE)	*Process as Processors*	Set of **platforms**, each supporting an assembly of components.

SCIPIO separates different aspects of the work.



Separation of project activities

We define project activities as modules so that the complexity of information and techniques can be encapsulated within a module.

Project activities have the following characteristics:

- A type of work within one or many projects.
- May be the responsibility of an individual or subteam within the project.
- May be performed as a set of tasks, as scheduled by the project manager.
- May deliver a series of products and services to the project as a whole, or to other project activities.
- May possess information and techniques that are hidden from other project activities.
- May be supported by independently deliverable skills and tools.

Project activities are designed for asynchronous parallel operation. This allows for the work to be distributed among several teams with different capabilities and/or interests, even at different locations.

Separation of workflow from processing, to enhance flexibility and control

An activity has what we might call an internal logic and an external logic. The internal logic of the activity specifies how it gets from a prior state or precondition to a posterior state or postcondition. The external logic specifies how the precondition

results from one or more prior activities, and how the postcondition enables one or more posterior activities.

A component may be used anywhere that its preconditions are satisfied. This means it doesn't need to know what the prior or posterior activities are, nor where they are located. Such components may be strung together, using separate workflow management software or middleware, to produce highly flexible and rapidly reconfigurable applications.

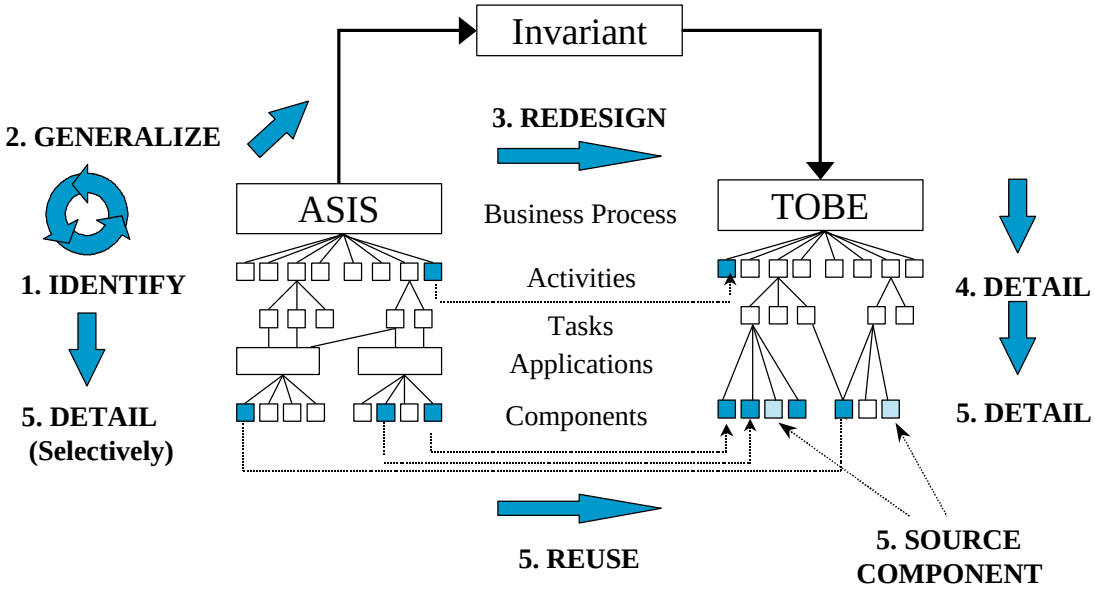
Thus while the internal logic is coded into the component, the external logic is coded into a separate piece of software, which serves as the workflow engine. It is this separation that results in enhanced flexibility.

Meanwhile, the workflow engine may monitor the progress of individual work packages as well as measure the throughput and performance of individual people or processors. If a work package gets held up somewhere, the workflow engine can make this visible. It is this monitoring function that results in enhanced control of the business process.

Task Structure

Analysis and design of a business process improvement

SCIPIO combines top-down and bottom-up analysis - we call this “middle-out”.



In rough terms, the steps are as follows.

1 Individual activities are understood through a process of investigation and communication with those who undertake the activity.

2 When sufficient information is available the individual activities are synthesized into a holistic and technology independent view of the business process.

3 Changes to the process are designed bearing in mind available technology and assumed possibilities for reuse. The new process design is recorded in a technology free manner.

4 The process is mapped to the underlying components by a process of decomposition into sub-processes, tasks, manual procedures and software components.

5 As the refinement of the new process proceeds validation is sought for assumptions about reuse of legacy systems. Candidates for reuse are selectively investigated in sufficient detail to establish what role they might play. As candidates are confirmed the design is adjusted to accommodate them. As this is going on component libraries both external and internal are also investigated.

The modelling and design work does not go strictly top-down, from broad structures to detailed specifications, nor strictly bottom-up, from detailed specifications to broad ('strategic') summaries, but middle-out. This means that each task or subtask may be executed at several levels in parallel.

SCIPIO offers a range of project scenarios.

Business Process Simplification Using IT to eliminate redundant steps in a business process, or to reduce excessive variety.	**Business Process Integration** Using workflow management tools and new software components to link business processes together more effectively.	**Business Process Transformation** Radical BPR
Application Simplification Using CBD in a tightly focused way to eliminate specific problems in legacy systems.	**Application Integration** Using middleware to link information systems together more effectively.	**Application Transformation** Building new information system solutions using commercially available components.
Technological Simplification Replacing unnecessary complications or variations in the technological infrastructure.	**Technology Integration** Linking heterogeneous technologies together more effectively.	**Technology Transformation** Satisfying new technical requirements, or exploiting new technological opportunities.

Modes of project activity

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.

You can use all of SCIPIO some of the time, and some of SCIPIO all of the time.

Foundation

SCIPIO is based on industry best practice.

Business Process Improvement	Case Working	
	Relationship Management	http://www.brl.com/
	Process / Workflow Management	http://www.aiai.ed.ac.uk/WfMC/
Business Excellence	EFQM	
	Baldrige	
Component-Based Development	CBD Forum	http:/www.butlerforums.com/
Object Modelling	Unified Modelling Language (UML)	http://www.rational.com/
Open Distributed Processing	ISO 10746	http://www.iso.ch:8000/RM-ODP/
Rapid Development and Deployment	Dynamic System Development Method (DSDM)	http://www.dsdm.org/

SCIPIO adheres to DSDM project management principles.

➢ Active user involvement in rapid application development is imperative.	➢ All changes during development are reversible.
➢ DSDM teams must be empowered to make decisions.	➢ Requirements are baselined at a high level.
➢ The focus is on frequent delivery of products.	➢ Testing is integrated throughout the lifecycle.
➢ Fitness for business purpose is the essential criterion for acceptance of deliverables.	➢ A collaborative and co-operative approach between all stakeholders is essential.
➢ Iterative and incremental delivery is necessary to converge on an accurate business solution.	

New techniques can be combined with previous techniques.