White Paper

Integrating the Soft Systems Process 'Stages' with EA Frameworks

WP0165 | October 2014

Ceri Williams

Ceri has thirty years in the IT industry, originally delivering complex control systems and subsequently broadening focus to Enterprise Architecture, Governance and transformation of the IT function. Working as a chief architect, consultant and coach, he enables FTSE 250 organizations to make medium and long term decisions on the shape of the Enterprise Architecture and positioning of the IT function.

He advocates putting people at the heart of technology and business change with focus on the human enablers and constraints. His work deals with the way in which rigorous engineering and architecture disciplines are integrated with the cognitive and behavioral capabilities of the people who practice them.

The previous white papers in this series [Ref 1] focused on the general features of SSM and how they differ and complement those of traditional engineering-inspired methods and frameworks. They considered how SSM is inclusive of all areas of the situation/ action space (i.e. scientific, technological, mechanical, material, as well as psychological, social and cultural), while an engineering approach excludes psychological, social and cultural influences. They described how an Enterprise Architect can appropriate elements of SSM and related social and cultural disciplines and blend them in as a defined part of a holistic approach to Enterprise Architecture.

This fifth paper in the series starts to explore the processes, steps and stages that the Soft Systems Methodology provides to guide practitioners, and the way in which they afford well-formed integration points for blending with engineering disciplines such as INCOSE and TOGAF. This paper sets the scene for the papers that follow to explore what SSM can add to commonly encountered Architecture Methods to enrich them and make them more effective.

There's no substitute for reading the papers themselves, but for readers short of time, the next section is an extract taken from Paper 1. It provides a very short outline of the Soft Systems Method - what it is, where it came from, and why it is significant. Readers wishing to deepen their background in the topic before embarking on this Paper can read the previous papers [Ref 1]. Readers already familiar with these papers can skip the next section.

A (very) Short History of Soft Systems

In a nutshell - the Soft Systems Methodology (SSM) is a systemic approach for tackling real-world problematic situations. Soft Systems provide a framework for users to deal with the kind of messy problem situations that lack a formal problem definition. Enterprise Architecture deals with “real-world problematic situations” and routinely encounters “messy problem situations that lack a formal problem definition” – this is why a re-imagining of Enterprise Architecture as a blend of Soft Systems and Systems Engineering disciplines is now needed. This blend provides us with a complete set of concepts and tools with which to operate in a complex, people-centric environment.

The Soft Systems Methodology originally emerged in the 1960s in response to problems encountered in tackling management and organizational problems using a systems engineering approach. From Ref [3]: “…the pattern of activity found in Systems Engineering – namely, precisely define a need and then engineer a system to meet that need using various techniques – was simply not rich enough to deal with the buzzing complexity and confusion of management situations”. I would add that the Systems Engineering approach also makes a number of (usually unstated) assumptions. Specifically that:

1. The problem and solution space can be modeled as a single definitive version of ‘the truth’ that is common to all stakeholders
2. A stable snapshot of the environment (people, process, material) can be baselined and persists largely unchanged during engineering analysis and solution delivery
3. The time taken to assemble the baseline and develop a solution is short enough that the solution is relevant and valuable at the time it is implemented

Every movement has its gurus, and Soft Systems is no exception. The first mainstream work to encode and specialize the knowledge around Soft Systems centered around Lancaster University, UK in the mid-1960s pioneered by Professor Gwilym Jenkins and subsequently by Dr Brian Wilson, before reaching the mass market through the work of Professor Peter Checkland. A number of useful references are included at the end of this White Paper.

Despite the name, the Soft Systems Method does not differentiate between ‘Soft’ and ‘Hard’ systems. It does not even treat ‘Hard’ and ‘Soft’ as features of the problem under consideration – they are features of the relationship between the problem and the person interested in it. They relate to the way in which the problem analyst perceives and interacts with the situation. For this reason it provides the best reference point for Enterprise Architecture and an inclusive, systematic framework for integrating Engineering and Soft Systems approaches. For the sake

of clarity in this series of papers, provided we accept that we construct our viewpoint to represent a ‘system’ and that ‘Hard’ and ‘Soft’ are not intrinsic to the system, we shall refer to ‘Hard’ and ‘Soft’ Systems.

For further reading and a very concise and complete account, see [Ref 2].

Key Concepts

For the purpose of this series of White Papers and in line with the general consensus in the field, Soft Systems and Hard Systems are treated as views of a system, rather than features of the system itself. Hard Systems are generally well suited to treatment with a Systems Engineering approach, soft systems with Soft Systems Methods. These viewpoints can be differentiated as described in Figure 1. The following Table 1 considers the main distinctions between Hard and Soft systems.

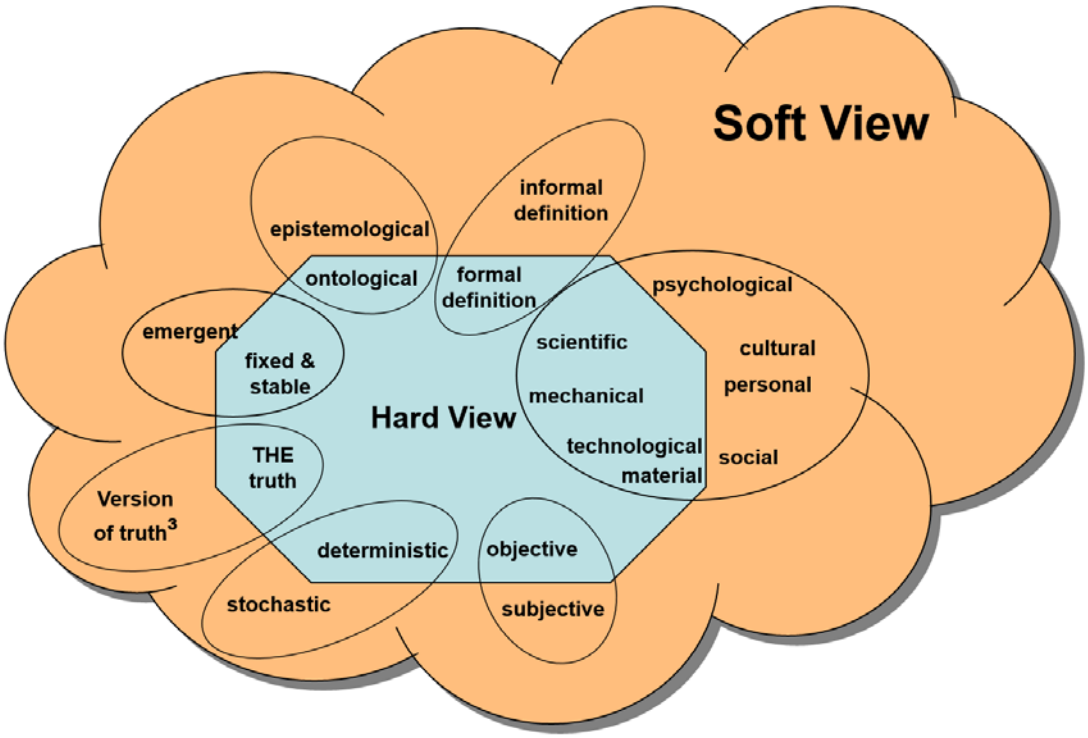

Figure 1 – The Relationship between Soft and Hard System viewpoints

#	Soft System View	Hard System View
1	Inclusive of scientific, technological, mechanical, material, psychological, social and cultural domains.	Inclusive of scientific, technological, mechanical, material domains. Exclusive of psychological, social and cultural domains.
2	Accepts that Systems develop emergent properties that cannot be foreseen at the outset. Provides concepts and tools to cater for this.	Assumes fixed and defined System and environment in which it operates. Unanticipated changes to either require re-entry into the Systems Engineering process at some point.
3	Provides the ability to integrate Systems that exhibit features and behavior that may be random, stochastic (i.e. statistical) and deterministic (i.e. individual cases predictable by analysis).	Deals effectively with deterministic systems and environments in which they exist. Has limited ability to deal with stochastic systems.
4	Tolerant and accepting of subjectivity and multiple 'versions of the truth'. Treats all models as viewpoints that express how stakeholders perceive the system. Accepting of dissonant and inconsistent viewpoints.	Considers multiple viewpoints as filtered views of a single, objective, canonical definition of a system or problem. Assumes and requires common agreement across all stakeholders, convergence and consistency of viewpoints.
5	Conceives of 'System' as an epistemological entity – i.e. as made up of conceptual and mental schemas and models that determine the perception of what the system is. Considers the perceptual schemas are an integral part of the 'system'.	Conceives of 'System' as made up of ontological entities – i.e. representation of, or actual entities physically existing or proposed to exist in the real world. The 'system' is independent of the way in which it is described.
6	Integrates Systems and problems that can and cannot be represented by formal definitions. Formal definition may not be possible either because of the nature of the System or because there is no suitable formal language with which to describe it.	Requires that problems and Systems can be represented by formal definitions (i.e. having conventionally recognized form, structure or set of rules). Assumes that they are structured, well-formed and logical.
7	Recognizes the significance of stakeholder values and world views (Weltanschauung) and their impact on the scope and shape of the System.	Recognizes stakeholder values and world views only to the extent that they filter the information that represents the system and separates stakeholder concerns.
8	Seeks problem and 'solution' definitions, actions and commitment to change that stakeholders can live with, rather than that they all agree on. SSM calls this 'Accommodation' between differing views.	Seeks consensus across stakeholders and requires that they believe the same 'truth'. Treats alternative views as incorrect and in need of change.
9	Inclusive of change to structures, processes and attitudes as a means of delivering improvement to a situation.	Inclusive of structures and processes, does not cater for attitudes.

Table 1 – Differences between Soft and Hard Systems Viewpoints

Soft Systems Processes

This paper assumes that the reader will be familiar with structured approaches to Enterprise Architecture and other similar analytical & engineering disciplines. All the industry favourites such as TOGAF, MoDAF, INCOSE, ITIL & DSDM Atern all make intensive use of key concepts, artefacts, processes and procedures. This rigorous approach is helpful in defining the frameworks in such a way that they lend themselves to practical implementation and inter-framework integration.

The Soft Systems Methodology pre-dates all of these and leaves more room for ambiguity and interpretation by the practitioner – although that's not to say that the industry favourites listed above are by any means free from ambiguity. SSM also focuses on process steps. One important enhancement to the common approach to structuring a method, is to explicitly make the practitioner a Methodologist. The dictionary definition of Methodology is "the study of Methods". SSM includes the development of the method by which participants organize their effort as an integral part of the method itself.

What appears slightly mind-bending at first, is in fact familiar as common practice among Enterprise Architects. It is not uncommon for an EA to have to find a way to link a system delivery method such as Agile/ DSDM with a strategic planning method such as TOGAF. It is also not uncommon to find a way to link one EA framework with another. This demand for method integration is partly a result of a sort of 'best of breed' approach, partly because stakeholders come from different worlds (e.g. Service Management & Operations will be familiar with ITIL) and partly because due to the force of circumstance, such as a merger or outsourcing decision, where different worlds are compelled to collide.

The traditional approach to integrating EA and System Delivery Frameworks is an analytical one, typically involving the cross-referencing and mapping of Roles, Processes and Products. This sort of approach is good as far as it goes. However, if you want the method to be inclusive of psychological, social & cultural features, the method integration will need to be rather more sophisticated. This paper proposes that if SSM becomes the 'hub', then all other methods can be related, interact through it in terms of People, Process and Product and be enriched by the availability of cultural features including: beliefs, values & behaviour norms (see Paper 4).

Figure 1 illustrates this as a 'hub and spoke' model, very similar to an efficient model for Information Systems integration – this works just as well for concepts and methods as it does for the exchange of digital control signals and data between IT systems.

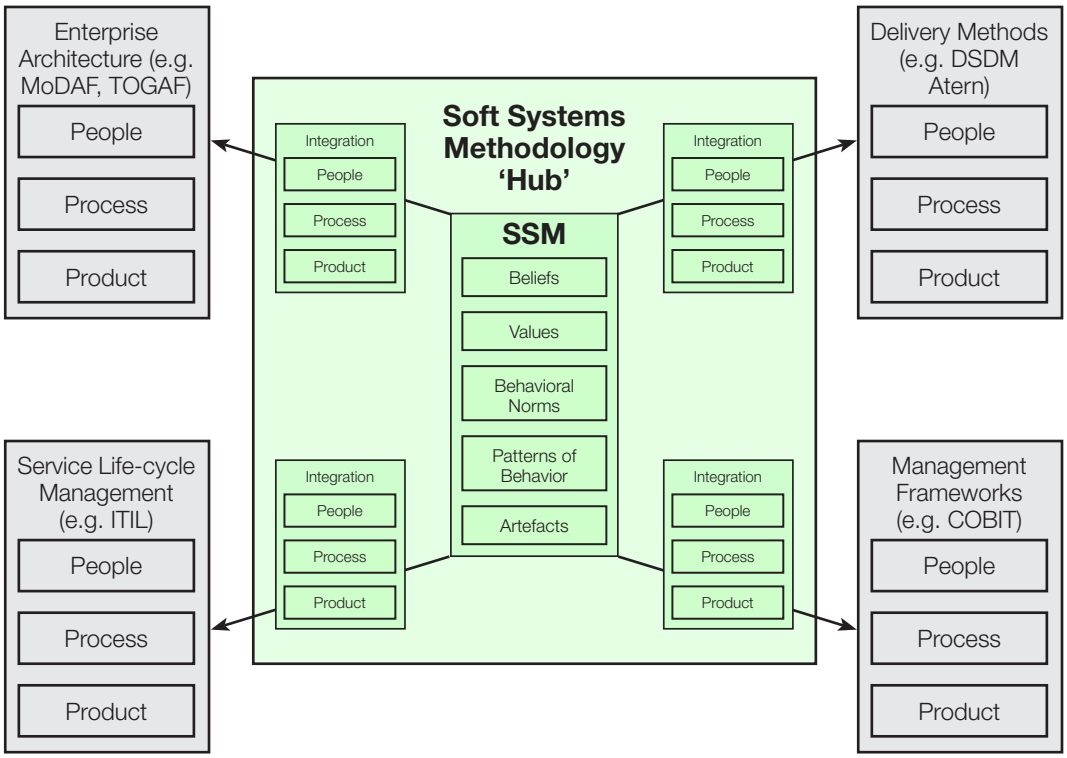

Figure 2 – Method Integration

Further exploration of this proposition requires some more detailed consideration of the processes and 'Steps' involved. SSM recognizes that most people are hard wired to be able to work with the idea of 'step by step' instructions – this is a pretty universal capability that applies as much to assembling an IKEA chair as building a target architecture. The art of judgment when applying these steps lies in how much detail and precision is of value, and how to manage iterations and feedback between the steps – and, critically, when to stop and move on. Figure 2 provides an outline of the principal SSM 'Stages'.

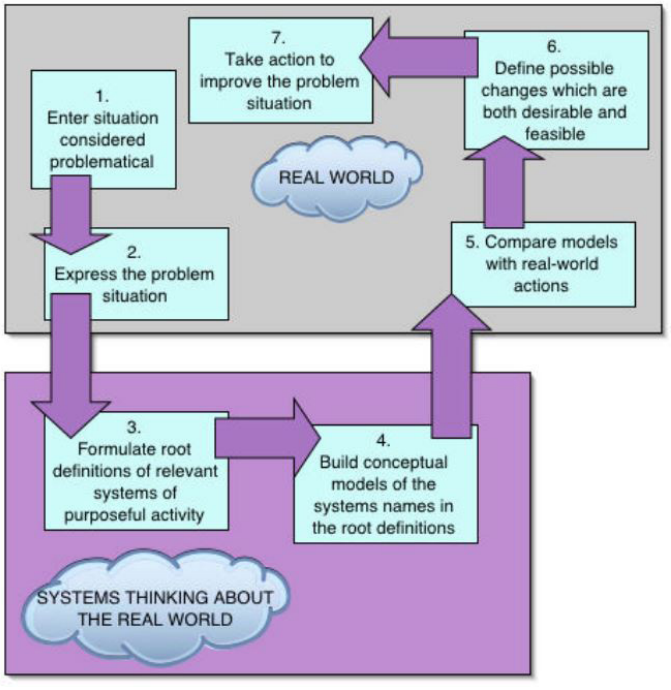

Figure 3 – SSM 'Stages'

The engineering mind will be tempted to interpret the flows in Figure 2 as a sort of sequential algorithm that is predictable and contains deterministic outcomes. There is some value in this in terms of organizing the activity needed to undertake an SSM project, but adherence to the activities and flows as anything other than a reminder of general principles of the method would build in inflexibility and undermine its value. From Ref [3] *"Speaking logically, then, SSM articulates a process of organized finding out about a problem situation, the finding out then leading to taking deliberate action to bring about improvement in the situation."*

Paper 1 made an initial attempt to map SSM Stages to TOGAF – in the light of subsequent experience, this now seems over-simplistic, but worth re-iterating, framed more as the alignment of 'centers of gravity' rather than a neat mapping along activity boundaries:

Soft Systems methods	Systems Engineering/TOGAF
Enter the problem situation	Feasibility Study & Concept Exploration/ Preliminary Phase
Express the problem situation	Concept if Operations/Architecture Vision
Formulate root definitions of relevant systems	System Requirements/Conceptual Architecture (Business, Information Systems & Technology)
Build conceptual models of (human) activity systems	High Level Design/Logical Architecture
Compare the models with the real world	Optioneering & Tradespace Exploration/ Opportunities & Solutions
Define changes that are desirable and feasible	Option Selection/Migration Planning
Take action to improve the real world situation	Implementation/Implementation & Governance

Table 2 – Multi-step Processes

The principal 'Steps' making up SSM are summarised below:

1. **Enter the problematic situation:** this involves acceptance by the participants - that they are prepared to assume responsibility for understanding a situation and working through improvements to it. It is important at this stage for participants to consider carefully what they can accept or must practically reject.
2. **Express the problematic situation:** involves an exploration of the situation and initial capture of that understanding, often in the form of 'rich pictures' and the beginnings of the conceptual models. Decisions at this point are made on what to include and exclude, and these decisions are captured explicitly.

3. **Formulate root definitions of relevant systems of purposeful activity:** consists of the explicit identification and capture of the relevant systems that carry out the purposeful activity of the situation. Typically these definitions would be shaped through consideration of: customers & stakeholders, actions & activities, transformation processes, world views, owners and environmental constraints (or CATWOE for short).

4. **Build Conceptual Models of the systems named in the root definitions:** further develops the root definitions to assemble the verbs describing the RD activities and structuring them to account for their relationships and dependencies. These models represent both operational features of the systems, as well as the 'meta-system' needed to monitor and adapt them to environmental changes.

5. **Compare the models with the real world:** more than just a 'gap analysis' of algorithms and data structures, this covers the rich set of features that SSM supports in modeling – including beliefs, values and behavioral norms.

6. **Define changes that are desirable and feasible:** a key part of this Step involves the participants working with each-other to determine what is important, and how different and similar their world views are. The aim here is to reach accommodations rather than consensus (see Paper 4).

7. **Take action to improve the real world situation:** focused on taking action and learning from it – where the learning may generate a revisiting of any or all of the previous stages. The participant's perception of the system will always significantly change by taking action to change it as intended and as unintended consequences become apparent.

One of the most difficult conversations with any sponsor who provides money, time, and energy for an SSM project is that there are no guarantees on how long it will take and what the outcomes will be. That said, I would argue that SSM is really just being transparent here about challenges that apply to all other engineering approaches, but about which they say little – they even create an illusion and expectation of predictability and repeatability that is very rarely justified. Metrics for Enterprise Architecture development and implementation are very, very rare. Even metrics for software and systems development are generally patchy and poorly formed, even thirty years after formation of the Function Point Users Group (see Ref [7]) and COCOMO (see Ref [8]).

SSM, like any other method that proposes implied or explicit steps, benefits from implementation that tackles engineering conservatism and the mind-set that prefers the comfort of a linear sequential process, to an interactive one where feedback is incorporated. While not part of the mainstream framework, SSM can learn something from software

engineering – iconically captured by Barry Boehm in his spiral model for software development, later revised as part of the Incremental Commit (ICM) model (see Ref [5] and [6]):

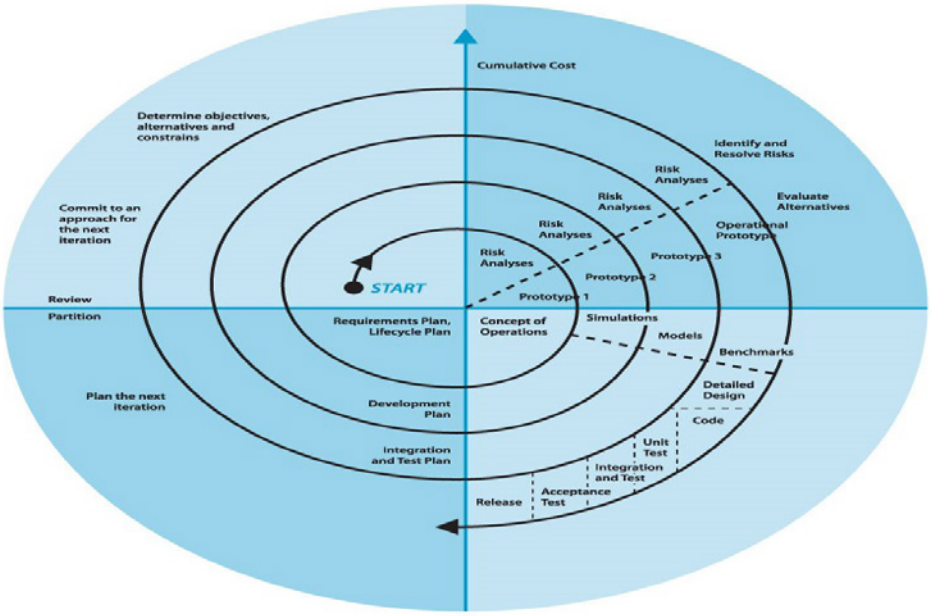

Figure 4 – Barry Boehm's Spiral Model of Development

The significance of this for the SSM movement is that each of the six Steps can be worked through quickly as a single iteration of the spiral, to provide a closed-loop learning cycle upon which to base the next cycle – and so on. SSM is particularly well adapted to enable such an implementation because learning and exploration is built in to every step – it embraces the flux of ideas and tolerates the disturbance that results from emergence of new information and perceptions. The engineering mind-set applied to Enterprise Architecture, on the other hand – especially when up against time constraints – tends toward locking down the Step prematurely. Freezing a target architecture and embodying it in an EA Tool repository while stakeholders are still interacting with it just stores up trouble for later and pretty much guarantees the model will not persist.

Applying the spiral model to SSM might look something like this:

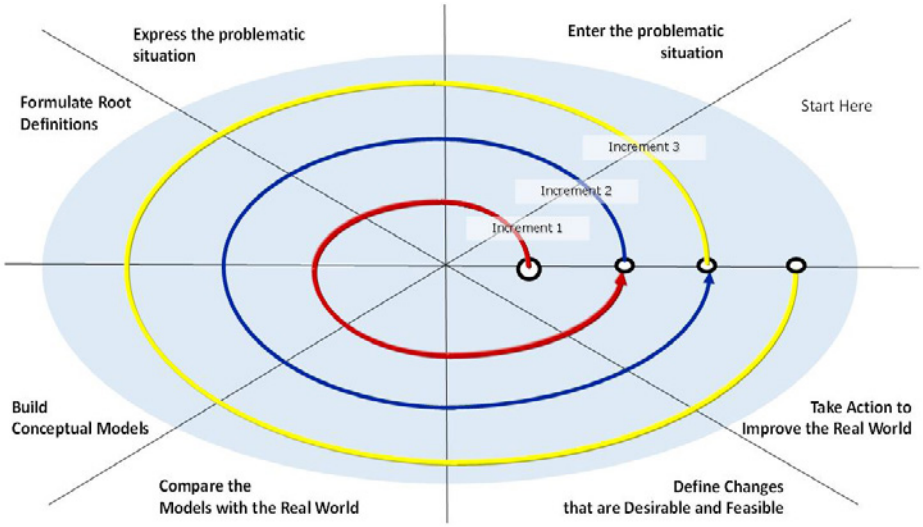

Figure 5 – The Spiral applied to SSM

The reasons a spiral model works well for SSM are similar to the reasons it works for software. As Barry Boehm observed, any linear or sequential process is based on a number of assumptions:

- The requirements are known in advance of implementation.
- The requirements have no unresolved, high-risk implications, such as risks due to cost, schedule, performance, safety, security, user interfaces, organizational impacts, etc.
- The nature of the requirements will not change very much during development or evolution.
- The requirements are compatible with all the key system stakeholders' expectations, including users, customer, developers, maintainers, and investors.
- The right architecture for implementing the requirements is well understood.
- There is enough calendar time to proceed sequentially.

Even in the real world of Systems Engineering and Enterprise Architecture, it is rare for the Engineer or Architect to encounter these conditions, rarer still for them to persist long enough to support any form of strategic planning. Recognizing these early and adjusting the EA approach to cater for them ensures that the Enterprise Architect focuses on defining an Architecture that has the best chance of being implemented.

White Paper #6:

White Paper #6 deep-dives into Steps 1 (enter the problem situation), 2 (express the problem situation) and 3 (Formulate root definitions of relevant systems of purposeful activity) and continues to explore the structured approach that the Soft Systems Methodology provides to guide practitioners, and the way in which this affords integration points for blending with engineering disciplines and frameworks.

I hope you have enjoyed this White Paper. Please get in touch if you have views to offer on the topic and feedback on the series, either direct to Orbus or via my email at: ceri.williams@theintegrationpractice.co.uk.

References:

[1] Enterprise Architecture meets Soft Systems Series, Papers 1-3. Orbus: www.orbussoftware.com/resources/authors/ceri-williams/

[2] Checkland, P & Poulter, J: learning for Action – A Short Definitive Account of Soft Systems Methodology and its use for Practitioners, Teachers and Students. ISBN: 9780470025543

[3] Checkland, P: Soft Systems Methodology www.yhcsleadership.co.uk/download-file/43

[4] Rosenhead, J (Ed): Rational Analysis for a Problematic World – Problem Structuring Methods for Complexity, Uncertainty and Conflict. John Wiley & Sons ISBN-10: 0471495239

[5] Boehm B, "A Spiral Model of Software Development and Enhancement", IEEE Computer, IEEE, 21(5):61-72, May 1988

[6] Boehm B, The Incremental Commit Model: http://ieee-stc.org/proceedings/2007/pdfs/BB1686.pdf

[7] International Function Point Users Group: www.ifpug.org/about-ifpug/about-function-point-analysis/

[8] The Constructive Cost Model (COCOMO): http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html

Orbus Software
3rd Floor
111 Buckingham Palace Road
London
SW1W 0SR
United Kingdom

+44 (0) 870 991 1851
enquiries@orbussoftware.com
www.orbussoftware.com

orbus software