Meeting Critical Real-World Challenges In Modelling Complexity: What System Dynamics Modelling Might Learn From Systems Engineering

Alan C. McLucas and Michael J. Ryan

University College, University of New South Wales
Australian Defence Force Academy
Northcott Drive, CAMPBELL 2600
AUSTRALIA

Telephone: +61 2 6268 8332 / +61 2 6268 8200
Facsimile: +61 2 6268 8276 / +61 2 6268 8443
Email: a.mclucas@adfa.edu.au / mj.ryan@adfa.edu.au

Abstract

System dynamics is still evolving. This paper argues additional rigour is needed if system dynamics is to achieve its full potential in helping us understand complex behaviour of human activity systems. It argues that a detailed appreciation of how systems engineers define, analyse, specify, manufacture, operate and support complex systems could inform the evolution of system dynamics even though there are significant differences between the two disciplines. The proffered approach integrates systems thinking, system dynamics modelling and systems engineering. This integrated approach enables group model building and building of exceedingly complex models through top-down design and careful management of the complexity introduced at each stage of the model-building process. The approach promises to engender greater confidence that models developed using it work and are both necessary and sufficient representations of the real world. The greatest potential gain accruing from application of this methodology is enhanced acceptance of system dynamics.

1. SYSTEMS ENGINEERING AND SYSTEM DYNAMICS MODELLING

In just over 40 years, system dynamics modelling has developed into a well-established body of knowledge. However, systems thinkers and system dynamics modellers are continually challenged to design and deliver highly effective strategies to remediate complex problems. How we go about building effective, verified and validated models of complex world behaviour tests our cognitive capacities to the limit. This paper suggests how we might exploit lessons and methodologies drawn from systems engineering practice to improve the effectiveness of system dynamics modelling, in particular by providing a framework within which we manage the complexity associated with modelling real-world behaviour.

1.1. A Common Interest in Understanding Complex Systems

The inescapable similarity between systems engineering and system dynamics modelling is that they both exist to help us understand complex systems. In systems engineering we use this understanding to define, analyse, specify, manufacture, operate and support systems whilst in

system dynamics modelling we set out to build models that assist us to manage within complex systems and to manage complex systemic problems. System dynamics modelling focuses on understanding complex systems and how they behave over time, using a single main technique of time-domain modelling and simulation supported by general systems theory and an appreciation of feedback structures.

1.2. Common Origins

Systems engineering (Faulconbridge and Ryan, 2003) methodologies and practices began to emerge from experience gained in the U.S. Department of Defense acquisition programs of the 1950s. These programs often involved complex and challenging user requirements that tended to be incomplete and poorly defined. Additionally, most programs entailed high technical risk because they involved large numbers of different technical disciplines and the use of emerging technology. Following a number of program failures, the discipline of systems engineering emerged to help avoid, or at least mitigate, some of the technical risks associated with the complex equipment acquisition programs. Systems engineering provides the framework within which complex systems can be adequately defined, analysed, specified, manufactured, operated and supported. Systems engineering processes and methodologies have continued to develop since the 1950s, and are widely applied to many of today's complex and challenging acquisition projects.

Within system dynamics modelling, feedback theory and cybernetics have strong links to engineering. The feedback theory of system dynamics modelling is profoundly important, having been drawn from engineering control theory (Richardson, 1990). Similarly, cybernetics, defined as the science of control and communication, in the animal and the machine (Wiener, 1948; Ashby, 1956) takes specific principles from engineering (feedback, stability, control, transmission and communication) and combines them with broader theories (requisite variety, bio-regulation and self-regulatory mechanisms) and uses them to formulate theories which we might apply to the general design of management and control in organisations.

Yet, whilst the feedback and cybernetic threads of systems thinking and system dynamics modelling contains abstractions drawn from engineering principles, system dynamics modelling is not commonly considered to be strongly related to systems engineering.

1.3. A Divergence in Focus

Despite their common origins, therefore, the disciplines of systems engineering and system dynamics modelling have diverged over the past forty years, for a number of reasons. In particular, systems engineering is seen to be applicable to technical systems involving *hard variables*, while system dynamics modelling is seen to be applicable to socio-technical systems involving *soft variables*.

A *hard variable* is one, which has attributes and relationships with other variables in a problem space to which physical laws apply. In the case of hard variables the *governing business rules* are readily formulated using numerical values and algebraic operators because these rules embody physical laws. Hard variables are readily quantifiable, and quantification can be verified. *Soft variables* are a class of variables, which includes a sub-class known as *intangibles*. Soft variables assist us in describing the complexity of human affairs in the context of human activity systems. In problem situations where soft variables apply, the governing business rules are not so readily formulated or formulated in a way which is faithful to real-world cause-and-effect they are intended to represent. They are not readily quantified.

Verification and validation of models involving soft variables are significantly more demanding than when models include only hard variables.

Because of its application to engineering problems, systems engineering tends to focus on engineering problems that are defined in terms of hard variables and solved with solid components, whether in hardware or software. On the other hand, system dynamics modelling tends to be applied to problems arising in social, socio-economic, and socio-technical systems, or biological, environmental or other systems involving people (or with which humans interact). Because of the integral role of human actors or interactions that humans have with parts of systems (such as environmental systems) most are considered to be purposeful human activity systems defined by soft variables (usually in combination with selected hard variables). Because the problems we address using system dynamics modelling are largely influenced by people, their mental models, their beliefs and values we are left to deal with the influences of soft variables much more so than systems engineers do.

It is very tempting therefore to consider systems engineering and system dynamics modelling to be applicable in different problem spaces. Before proceeding, we need to make a distinction between *system dynamics modelling* and *systems thinking* (or soft systems methodologies).

Wolstenholme (1990: 3) defines *system dynamics modelling* as:

> A rigorous method for qualitative description, exploration and analysis of complex systems in terms of their processes, information, organisational boundaries, and strategies; which facilitates quantitative simulation modelling and analysis for the design of system structure and control.

Checkland (1993: 318) defines systems thinking as:

> An epistemology which, when applied to human activity is based on four basic ideas: *emergence, hierarchy, communication* and *control* as characteristics of systems. When applied to natural or designed systems the crucial characteristic is the *emergent properties* of the whole.

Significantly, the system dynamics modelling definition has two major elements, the 'softer' method of qualitative system dynamics modelling (alternatively known as systems thinking or soft systems methodologies, designed to reveal the details of mental models held by stakeholders in a given problem), which informs the 'harder' form of quantitative system dynamics modelling. We are bound to include soft variables in combination with hard variables in system dynamics models because they are important. The only thing we can say with certainty about models of real-world problems involving human activity systems that do not include the effects (influences) of soft variables is that they will be wrong (Forrester, 1961: 57; Sterman, 2002: 523).

Sterman (2000: 37) argues that our mental models are dynamically deficient, that is, they omit feedbacks and time delays [and the consequences of system response], accumulations and non-linearities with the consequence that simulation is the only practical way of testing our mental models, noting that the complexity of [both the real world and] our mental models vastly exceeds our capacity to understand their implications. To build simulations we must construct system dynamics models built upon algebraic relationships and mathematical integrations. Therefore, even though a system dynamics model may include consideration of the influences of (or effects produced by) soft variables, it must ultimately become a hard

(quantitative) representation of a particular problem expressed in precise mathematical way.

Simulation performance of our models must be shown to be reliable, repeatable, and produce behaviours traceable back to real-world cause-and-effect. If the client has visibility of, or is involved in, the processes of building and testing models this can build shared understanding and confidence. When making the transition from conceptual systems thinking (soft systems) models to quantified system dynamics models we might be tempted to interpret the quantified model as the most plausible representation of the problem at hand. However, without a sound model-building framework that includes comprehensive testing, we cannot have confidence in the veracity of the transition from conceptual to quantified model or the quantified model itself.

Consequently, while systems engineering and the qualitative aspects of systems dynamics may be considered to occupy different parts of the problem space, systems engineering has much to offer in aiding us to design, build and test quantitative system dynamics models including the detailed transition from conceptual representation to quantified model. In the sections that follow it is argued that the most important gain to be made is in the area of confidence that quantified models we build will function as intended, and that testing verify and validate the model is both routine and comprehensively applied through systems engineering methodology.

2. SYSTEMS ENGINEERING METHODOLOGY TO ENHANCE SYSTEMS DYNAMICS MODELLING

There is a wide range of systems engineering definitions, each of which tends to reflect the particular focus of its source (Faulconbridge and Ryan, 2003; DSMC, 1990; EIA/IS 632, 1994; IEEE-STD-1220-1994, 1995; Sage and Rouse, 1999). Perhaps the most useful is: "…an interdisciplinary, comprehensive approach to solving complex system problems and satisfying stakeholder requirements" (SECMM-95-01, 1995). Although each of these definitions has a slightly different focus, a number of common themes are evident and are described in the following sections. Of particular interest to us here are the themes of requirements engineering, a top-down approach to managing and coping with complexity, verification and validation, and a mechanism for integrating many disciplines and specialisations.

2.1. Requirements Engineering

The complete and accurate definition of system requirements is the primary focus early in any systems engineering effort. The lifecycle of a systems design begins with a simple statement of need, which is translated into a large number of statements of requirement that form the basis for the functional design and subsequently the physical architecture. These transitions must be managed by a rigorous process that guarantees all relevant requirements are included (and all irrelevant requirements excluded). The establishment of a set of correct requirements is fundamental to the success of the subsequent design activities.

Once requirements have been collected, the systems engineering process then focuses on the management of these requirements from the system level right down to the lowest constituent component. This *requirements engineering* (sometimes referred to as *requirements management* or *requirements flowdown*) involves elicitation, analysis, definition and validation of system requirements. Requirements engineering ensures that a rigorous approach is taken to the collection of a complete set of unambiguous requirements from the stakeholders.

Requirements traceability is also an essential element of effective management of complex projects. Through traceability, design decisions can be traced from any given system-level requirement down to a detailed design decision (*forward traceability*). Similarly, any individual design decision must be able to be justified by being associated with at least one higher-level requirement (*backwards traceability*). This traceability is important since the customer must be assured that all requirements can be traced forward and can be accounted for in the design at any stage. Further, any aspect of the design that cannot be traced back to a higher-level requirement is likely to represent unnecessary work for which the customer is most probably paying a premium. Traceability also supports the change process, especially the investigation of the impact an intended change might have.

Support for requirements traceability is a feature of the top-down approach, one which is a consequence of taking a holistic view, that is, the unquestioned world view, *weltanschauung*, to which Checkland (1981) and Checkland and Scholes (1999) refer. This top-down approach provides a mechanism by which it can be guaranteed that requirements can be satisfied at any stage. A bottom-up approach cannot provide the same guarantee, nor does it enable the systematic discovery of emergent properties.

2.2. A Top-down Approach to Coping with Complexity

Traditional engineering design methods are based on a bottom-up approach in which known components are assembled into subsystems from which the system is constructed. The system is then tested for the desired properties and the design is modified in an iterative manner until the system meets the desired criteria. This approach is valid and extremely useful for relatively straightforward problems that are well defined. Unfortunately, complex problems cannot be solved with the bottom-up approach.

Systems engineering begins by addressing the complex system as a whole, which facilitates the initial allocation of requirements as well as the subsequent analysis of the system and its interfaces. Once system-level requirements are understood, the system is then broken down into subsystems and the subsystems further broken down into components until a complete understanding is achieved of the system from top to bottom.

This top-down approach is a very important aspect of managing the development of complex systems. By viewing the system as a whole initially and then progressively breaking the system into smaller elements, the interaction between the components can be understood more thoroughly, which assists in identifying and designing the necessary interfaces between components (internal interfaces) and between this and other systems and the environment (external interfaces). For example, Figure 1 illustrates the ANSI/EIA-632 (1999) approach to top-down development.

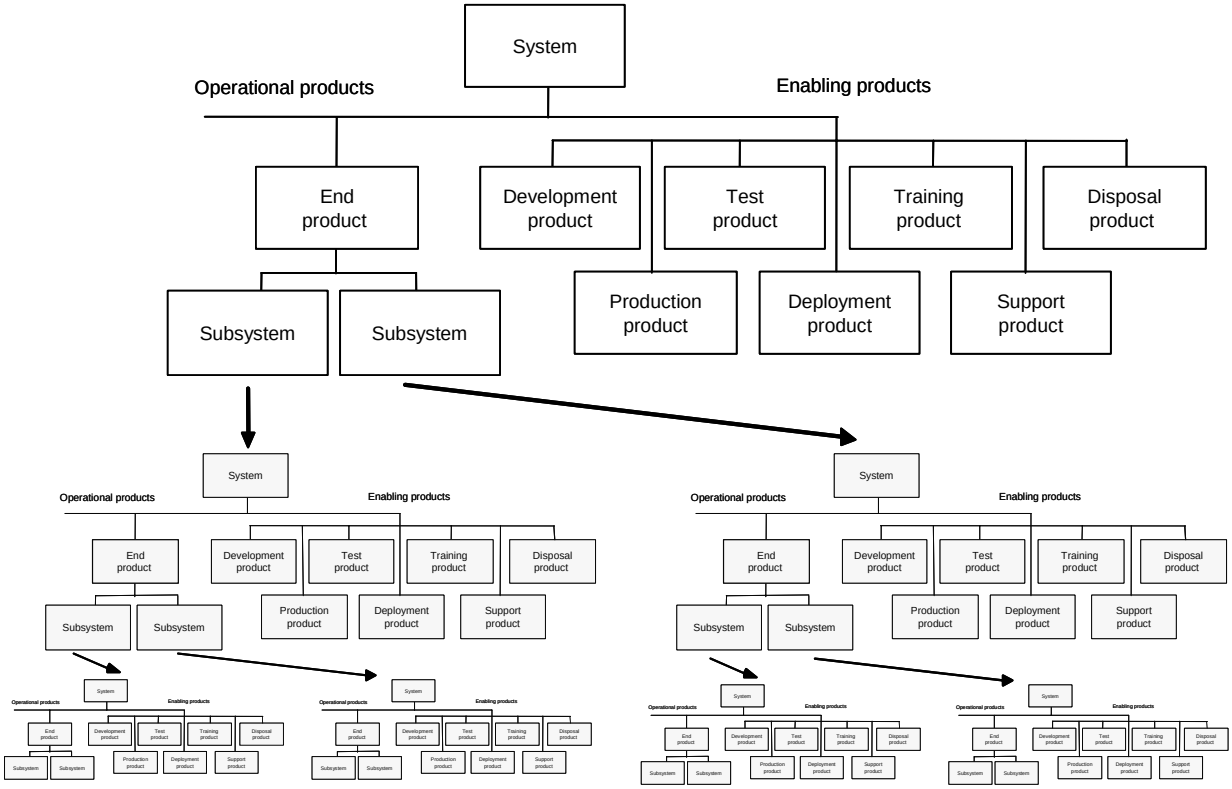

Figure 1. ANSI/EIA-632 building block concept for top-down development.

It must be recognized, however, that while design is conducted top down the system is implemented using a bottom-up approach. That is, one major aim of system engineering can be considered to be to provide a rigorous, reproducible process by which the complex system can be broken into a series of simple components that can then be designed and developed using the traditional engineering bottom-up approach. Importantly, the other major aim of systems engineering is to provide a process by which the components and subsystems can be integrated *(synthesised,* in systems engineering terms) to achieve the desired system properties.

Integration aims to combine lower-level components into progressively higher-level subsystems until the system is complete. While the design process has been conducted top-down, the integration process is conducted bottom-up using well-proven techniques. At each stage of the integration, some form of integration testing is conducted to verify the successful integration against the appropriate level of documentation. Eventually, when systems integration is complete, testing can be conducted at the system level against the original requirements. Test and evaluation plays a role in all phases of the systems engineering effort. The integration effort is summarized in Figure 2. Note that the terms *system*, *subsystem* and *component* are relative. Each system comprises subsystems that consist of components. Each subsystem, however, can be considered to be a system in its own right, which has subsystems and components and so on.

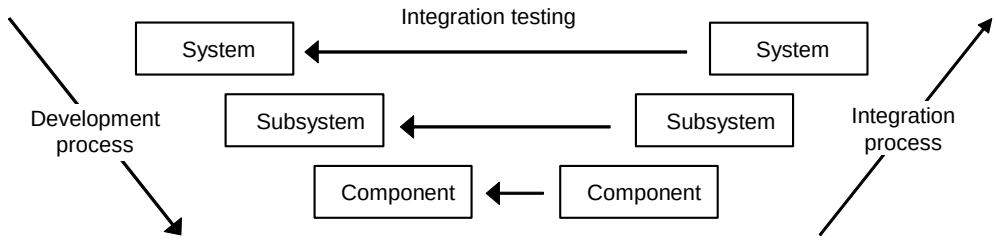

Figure 2. Top-down development and bottom-up integration process.

2.3. System Optimisation and Balance

During system design, it is important to remember that system performance is of vital importance rather than the performance of the individual subsystems and components (Faulconbridge and Ryan, 2003). It does not necessarily follow that the combination of optimised subsystems leads to an optimised system. Additionally, the system architecture must represent a balance between the large number of requirements that, as well as technical considerations, cover a wide range of factors such as environmental, ergonomic human factors, moral, ethical, social, cultural, psychological, and so on. This system optimization and balance (treatment at the most appropriate and consistently applied level of aggregation) can only be guaranteed using a top-down approach.

2.4. Integration of Disciplines and Specialisations

Systems engineering aims to manage and integrate the efforts of a multitude of technical disciplines and specialisations to ensure that all user requirements are adequately addressed. Rarely is it possible for a complex system to be designed by a single discipline or by selected individuals working alone. Consider an aircraft example. While aeronautical engineers may be considered to have a major role, the design, development and production of a modern aircraft system requires a wide variety of other engineering disciplines including electrical, electronics, communications, radar, metallurgical, and corrosion engineers. Of course, in system terms, other engineering disciplines are required for testing and for logistics and maintenance support as well as the design and building of facilities such as runways, hangars, refuelling facilities, embarkation/disembarkation facilities, and so on. Other non-engineering disciplines are involved in project management, marketing, finance, accounting, legal, environmental, and so on. In short, there could be hundreds, even thousands, of engineers and members of other disciplines involved in the delivery of an aircraft system.

The aim of the systems engineering function is to break up the task into elements that can be developed by these disparate disciplines and specialisations and then provide the management to integrate their efforts to produce a system that meets the users requirements. In modern system developments, this function is all the more important because of the complexity of large projects, their contracting arrangements and the geographic dispersion of contractor and subcontractor personnel across the country and around the world.

3. A MORE-RIGOROUS APPROACH TO SYSTEM DYNAMICS MODELLING

Systems engineering, therefore, has much to offer system dynamics modelling. In particular, systems engineering offers a framework through which design and development discipline is applied and rigour assures reliable outcomes. System dynamics models bring together hard and soft aspects that are difficult to evaluate and test (verify) in detail, but to assure the necessary rigour and opportunities to learn, comprehensive testing is essential.

Arguably, the challenges faced by system dynamicists in verifying their models (that they work 'right' in accordance with requirements) are greater because of the influence of soft variables and the variability of human behaviour in human activity systems and human response to changes (changes applied through exogenous forces or those which develop within the human activity system). This places greater demands on the modeller to be able to design and apply tests that verify that the models actually reproduce the cause-and-effect relationships of the real-world problem situation. System dynamics models must explain real-world behaviour through structure and equations that reflect real causal relationships as they appear in the real world system (Forrester, 1961: 115-129). These models must be tested to assure that they are the 'right' models (validated) to provide the detailed insights needed for the design and development of appropriate remedial strategies. So, discipline and rigour in system dynamics modelling is essential.

3.1. Requirements Engineering

There is considerable evidence that, from the viewpoint of cognitive capacity, we are poorly equipped to deal with complex problems (Diehl and Sterman, 1995; Dörner, 1980; Forrester, 1971; Kleinmuntz, 1985; 1993; Mosekilde and Larson, 1988; Mosekilde, et, al., 1990; Paich and Sterman, 1993; Sterman, 1989a; 1989b; 1989c; 1994; 2000; 2002 and Sweeney and Sterman, 2000). Where we might encounter large amounts of detail (*detail complexity*), we need tools and techniques to manage that detail. Tools such as databases and spreadsheets are widely used and we are generally familiar with these. It is a different story, however, when it comes to problems where we might encounter the additional aspect of *dynamic complexity*. That is, dynamic complexity in problems is characterised by time dependence, tightly coupled elements, feedback, non-linearity, history dependence, adaptability, counter-intuitive system response, policy resistance and trade-offs between short run and long-run remediation (Sterman, 2000: 22). We need specialised tools and techniques to help us build models, which create for us the best possible opportunities to understand these complex, dynamic problems. Again, rigour in model building supported by both verification and validation is essential. Systems engineering enforces that rigour through requirements engineering and requirements management and formal recognition of verification and validation within the systems engineering methodology.

System dynamics modelling supported by the tools and techniques of the systems engineering discipline offers to improve the effectiveness of our problem-solving approach. Noting that system dynamics modelling can be iterative (Forrester, 1975: 245; Homer, 1996) the proffered approach creates opportunities to reduce the number of iterations needed to build truly viable models, whilst enhancing particular opportunities to learn because modelling activities will be highly visible, comprehensively documented and explained.

It is acknowledged that focusing on model building (rather than on producing the model as an artefact) can be highly valuable in generating learning (Richardson and Pugh, 1981; Forrester, 1985; Morecroft, 1992; Morecroft and Sterman, 1992; Morecroft and Sterman, 1994; Homer, 1996; Sterman, 2000). It is also acknowledged that a model-building strategy having as its primarily goal the development of a working model might actually preclude certain opportunities along the way to experiment and learn *ab initio* about system dynamics structure and causality.

Clearly, there is a conflict here. This conflict could be resolved by reflecting on the main purpose of each particular model-building project. If the primary purpose is to maximise learning (by experimentation and progressive development and testing of dynamic hypotheses)

and ample time is available for this to occur, then a highly iterative model-building process would be in order. If the primary purpose is to produce modelling outputs through an efficient set of processes having learning as a secondary (but very important) purpose then an approach such as that described here, integrating systems engineering and system dynamics modelling approach, would be more appropriate.

Integrating the systems engineering approach into a methodology for building of system dynamics models for analysis of complex problems provides a number of benefits. The proffered approach integrates:

- *Requirements Analysis.* The complete and accurate definition of the systemic problem, demands primary focus on clear and unambiguous statements of requirements for a model of that systemic problem. Defining requirements forms the basis for the structural and physical designs of models, which will be used for analysis of the specific parts of a physical system or human activity system under study. The initially unstructured problem situation is translated through to precise definitions of the human activity system and its behaviour. These definitions are critical for many reasons including being the basis for model design, development and verification, that is, the detailed testing against requirements. To facilitate this analysis we might use Soft Systems Methodology (Checkland, 1981) to develop a series of root definitions of the relevant parts of the human activity system. A root definition is a concise, tightly constructed definition of a human activity system which states what the system is; what it does is then elaborated in a conceptual model that is built on the basis of the definition. Every element in the definition must be reflected in the model derived from it. A well-formulated root definition will make explicit each of the *Customer, Actor, Transformation, Weltanschauung, Owner,* and *Environmental constraints* (CATWOE) elements identified by Checkland (1981) and Checkland and Scholes (1999). Building root definitions of human activity systems raises the level of consideration of the problem to the system level (initially, at least). The advantage of such a system-level consideration is that it enhances top-down thinking (consistent with the systems engineering approach). This contrasts strongly with those system dynamics practices in which casual loop diagramming only informs bottom-up model building.

- *Requirements Management.* Once requirements have been collected, the systems engineering process then focuses on the management of those requirements from the highest level of aggregation right down to the lowest. It is appropriate to define requirements in terms of a model at the highest level, a sector at the intermediate level and a module at the lowest level. Modules contain elements such as stocks (levels or accumulators), rates (flows), auxiliary variables, physical flows and information links (the latter two often forming feedback loops). To collect and build specifications of modelling requirements involves activities of elicitation, analysis, definition and validation. Requirements engineering ensures that a rigorous approach is taken to the collection of a complete set of unambiguous requirements from the stakeholders in each of their perspectives (even though at some stage these will be combined into a single view represented by the model, or set of connected models).

- *Requirements Traceability.* Through being able to trace requirements back to stakeholders and through every stage of consideration of a complex problem, it is

possible to assure that all requirements can be traced forward and can be accounted for in the design of the model at any stage of its development. Similarly, any individual design decision affecting the model of the system, or the system itself, must be able to be justified by being associated with at least one higher-level requirement (backward traceability). Further, any aspect of the model being used to analyse the design, and the design itself, that cannot be traced back to a higher-level requirement is likely to result in redundant or unnecessary work. Traceability also supports processes that lead to implementation of changes to the structure of the systemic problem under examination. Support for requirements traceability is a feature of the top-down systems engineering approach that provides a mechanism by which it can be guaranteed that requirements can be satisfied at any stage. A bottom-up approach cannot provide the same guarantee.

Any model-building activity must be managed as a project designed to deliver specific outcomes. These outcomes include learning through experimentation with the model and delivery of a completed model. The project must be based on requirements:

- *elicited* with close engagement of key stakeholders, notably the client group for whom the model is being built;

- *analysed* for logical construction and completeness—here conflicts between requirements must be resolved whilst maintaining a minimum set of requirements which must be met (built into the model) according to priority;

- *defined*, that is, clearly and unambiguously specified using precise language;

- *validated*, that is, tested to determine the extent to which the requirements are likely to lead to development of a model which maps sufficiently onto the real-world problem space;

- *managed* throughout—here the modelling project must be managed in terms of its:

    * *scope*, particularly as the meanings of modelling requirements (interpretations of dynamic hypotheses) are questioned, which often leads to pressure to change the scope of the modelling effort ; and

    * *configuration*, that is, what form the model will take and what it will include or exclude must be constantly monitored through a set of formal processes, especially where the allocation of effort to tasks involves potential overlap and confusion;

- *traceable* through each step, including:

    * back to the original requirements (and testable, that is, verifiable against the original requirements); and

    * any changes that may have been permitted through configuration management.

The system dynamics modelling process is illustrated in Figure 3 (adapted from Forrester, 1994: 245), annotated to show where systems thinking and soft systems methodology and

systems engineering activities are integrated.

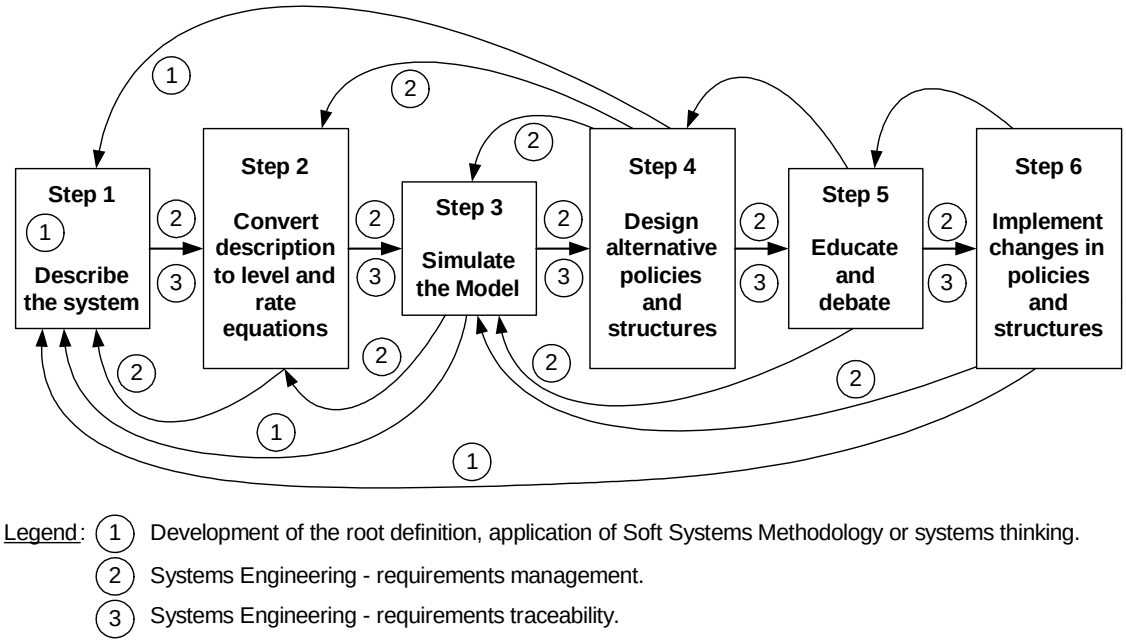

Figure 3. System dynamics modelling process annotated with Soft Systems Methodology, Systems Thinking and Systems Engineering activities.

3.2. A Top-down Approach to Coping with Complexity

3.2.1. Managing Complexity

Systems engineering begins by addressing the complex system as a whole, which facilitates the initial allocation of requirements as well as the subsequent analysis of the system and its interfaces. Once system-level requirements are understood, the system is then broken down into subsystems and the subsystems further broken down into components until a complete understanding is achieved of the system from top to bottom. This top-down approach is a very important element of managing the development of complex systems. The approach leads to clear definition of the component parts (modules), sectors and co-models and the interfaces between them (McLucas and Ryan, 2005). A generic diagrammatic representation of a module is the lowest-level component part, is shown in Figure 4 (McLucas, 2005).

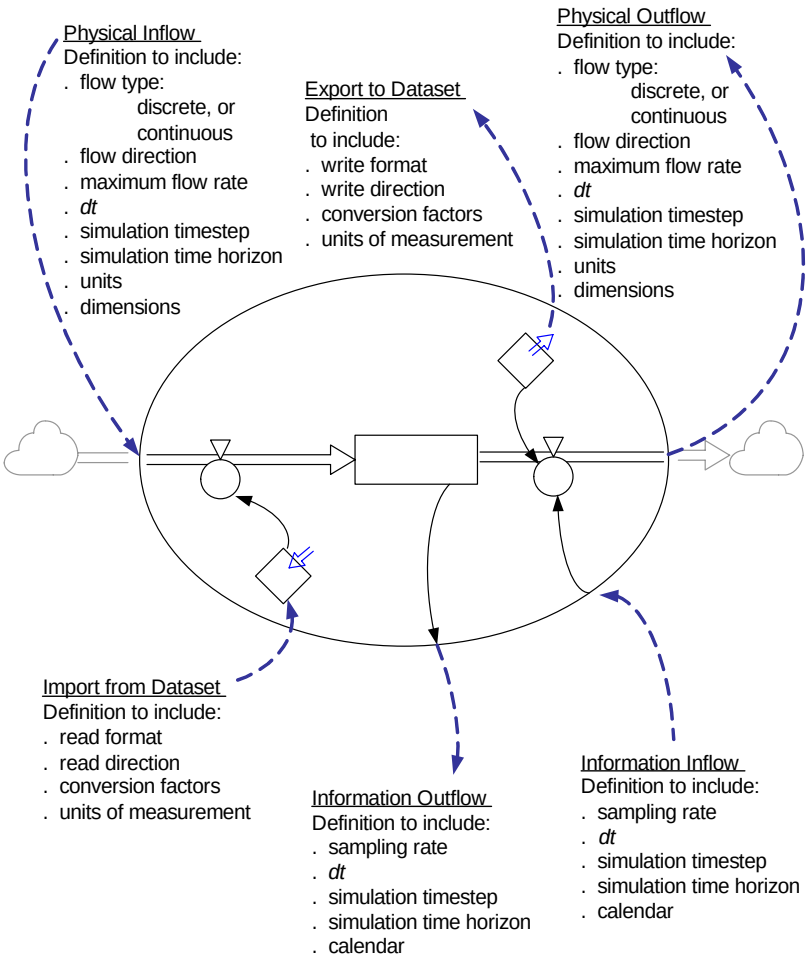

Figure 4. Generic system dynamics modelling module.

This approach results in creation of starting definitions of each module for a specific problem, such as that (by way of example only) explained by Sterman (2000: 285-289) as shown in Figure 5. This example considers how multiple feedback mechanisms, some including non-linearities (though that is generally not known at this early stage), can produce complex behaviour in populations where the environment has limited capacity to carry a population.

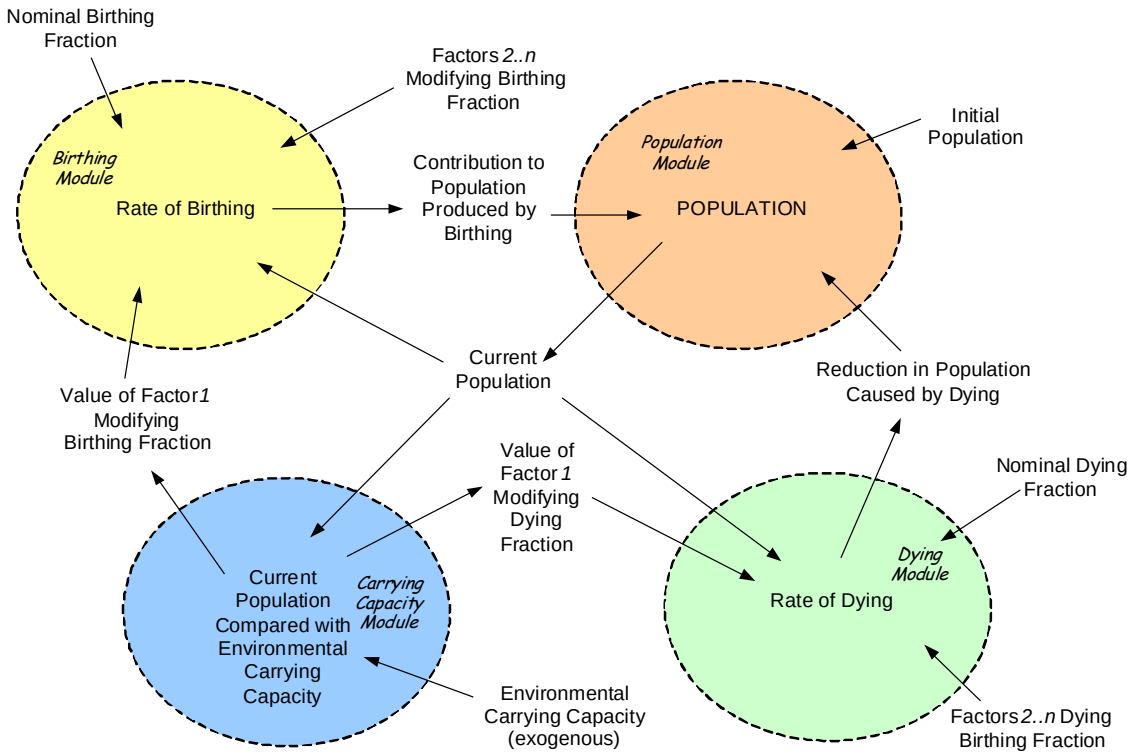

Figure 5. Module Formulation of Population Problem.

From a process point of view, requirements and design are approached top-down (with Figure 5 being an early artefact of the process) but detailed construction follows a bottom-up approach (once each functional module has been described, such as shown in Figure 4). Note that in the process, Figure 5 would be developed before Figure 4.

The subsequent building of each module, verification and their combining (through integration and synthesis) must be managed through systems engineering methodology, which has the rigour and discipline to assure that none of the system's functionality is lost and emergent properties are systematically discovered. The systems engineering approach also assures that processes of analysis, design and construction can be reproduced and can be implemented in a way that still enables use of the traditional bottom-up approach. It also enables the allocation of modelling effort to building, testing and subsequent synthesis in a way that avoids duplication of effort, misnaming of variables occurring at the interfaces between modules. Consequently, once functional requirements and modules have been defined, group model building becomes a routine matter.

Top-down analysis creates the framework within which bottom-up construction of modules and sectors and ultimately integration into models can then occur. This approach is facilitated by formulation of a clearly defined systemic structure which will lead to creation of models exhibiting the necessary system-level behaviour, that is, models which replicate the reference modes of behaviour (in systems engineering terms, delivers required functionality).

Graham (1977) noted that experienced system dynamicists perceive that situations that appear to be very different on the surface are caused by fundamentally similar mechanisms. This is the *structure* referred to by Forrester (1961: 2) and explained by Goodman (1989) and various teachers through to Sterman (2000). Many of the fundamental structural elements have

been defined by Hines et al. (1996; 1997; 2000), Coyle (1996) and McLucas (2005). However, *structure* involves more than archetypical behaviour and defined *molecules of system dynamics structur*e or *common modules*. It also encompasses the combined elements (such as feedback mechanisms, non-linear relationships and shifting feedback loop dominance), which produce complex dynamics behaviour (and, indeed, are emergent properties of such systems).

Explicit and directed study of these molecules and combined elements of structure are essential to developing in the student requisite skills in conceptualisation and development of statements of requirement needed for the subsequent building and testing of models. However, once the conceptual structure of the required model has been developed through a set of top-down procedures, it is essential to build the model through stepwise processes that at no time permits the complexity introduced to exceed our ability to understand. This should enable us to build models of problems that are exceedingly complex, and in theory at least have unlimited complexity.

The systems engineering approach also facilitates management of construction of the model in its most-highly aggregated form by enabling the group modelling leader having the overall responsibility for model construction to reflect upon the known behaviour of building blocks of structure. Feedback loops of archetypical system dynamics structures will frequently exist across the boundaries of the modules defined for the group modelling activity. Whilst the group modelling leader will be able to seek out instances of archetypical structures, he or she will be able exploit these structures by coordinating their linkage across the architecture and at each of the defined module / sector / co-model boundaries. The systems engineering approach has the added advantage that it routinely enables the discovery of those feedback mechanisms (and the emergent properties with which they are associated) even when those building individual modules, say as described in Figure 5, do not know explicitly that inputs or outputs to the module they are currently working on are part of a feedback loop.

To manage complexity, therefore, we need to be able to specify the requirement for our model, allocate these requirements appropriately to sectors and then, within sectors, allocate requirements to modules. Having developed and tested the modules, we integrate them into sectors, test the sectors against the subset of requirements, integrate sectors into the model and then test against the system-level requirements. The 'testing' (verification and validation) is discussed in more detail in Section 3.2.4.

3.2.2. Reducing Complexity

Managers often face socio-technical problems of *order* in the range 10^{th} to 100^{th} (Forrester, 1975: 66). It is acknowledged that *order* is only one indicator of complexity. This complexity easily outmatches our ability to reliably use judgement and intuition. Sterman (2000: 29) observes that our cognitive capability is barely sufficient to enable us to mentally simulate a first-order linear positive feedback system. Using the Complexity Index C defined by Kline (1995) such a system would be defined by $C \approx 4$. The problems of interest to system dynamics modellers, such as those identified by Forrester (1975) have complexities characterised by $C >> 10^6$. Kline suggests real problems we face can be characterised by values C in the range 10^9 to 10^{13}, noting that for these problems C is the product of the number of state variables, the number of independent parameters describing the problem and the number of feedback loops (endogenous feedback loops plus those which cross the boundary of the problem space, linking to exogenous factors). We need effective ways of reducing the complexity encountered as we progress through each stage of model building, even though we aim to build models of highly complex problems.

A study (McLucas and Ryan, 2005) of 30 models published in *System Dynamics Review*, since 1985 (for which full code listings were available), each held up as exemplars of system dynamics modelling practice, suggests that:

- we successfully and consistently use system dynamics modelling to analyse problems having complexity 10–1,000 times our inherent cognitive capability, the complexity modelled having a mean of 100 times; and

- these models typically contain 3–20 modules (the mean being nine), where we have the cognitive capability to analyse each in isolation.

Interestingly, without formally observing any stated systems engineering methodology, it appears that the experienced builders of the models sampled coped intuitively with the complexity of the problem being modelled by breaking the task down into modules of manageable complexity. Because most of these models were built using software applications which demanded high levels of skill in writing code, discipline in model construction was essential (particularly in formulation of blocks of code for functional modules). While we might expect such intuition from experienced modellers, a more-formal approach is necessary to ensure that top-down decomposition is a mandatory element of system dynamics modelling, particularly when being taught to, and applied by, novice modellers. Arguably, this is more important where object-oriented system dynamics software applications are commonplace and building fully 'wired-up' models is quick and easy, and potentially erroneous as a result.

3.2.3. Emergent Properties

Traditional problem solving involves working from the bottom up. The bottom-up approach assembles well-known, well-understood and manageable components into subsystems. Emergent properties therefore cannot be predicted solely by looking at the components (Stevens, et al., 1998: 94). A bottom-up approach does not deliberately enable the discovery of emergent properties. In system dynamics modelling, analysing reference modes of behaviour as part of a top-down approach (consistent with systems engineering) systematically aids discovery of one particular form of emergent property. This is complex dynamic behaviour produced by feedback and delay. Without this aspect of methodology, the emergent properties and the real causes for them being produced may remain undiscovered.

3.2.4. Verification and Validation

In engineering there is a very strong link between the model and causal explanations underpinning the model, as evidenced by physical laws and the emphasis placed by Engineering Faculties on studying those physical laws. In systems engineering it is expected that these causal explanations can be 'proven'. In system dynamics modelling, despite the warning provided by Forrester (1961: 115-129) that we must explain real-world behaviour through the structure and equations which reflect the real causal relationships in the real-world system, examples of system dynamics models which mimic the real world (but for which there is no real 'proof') can be found. Unfortunately, the consequence is poorly built models in which we can have little confidence (though they might be 'sold' to clients as affording powerful insights into their problems). The reasons for this are neither trivial nor does it necessarily suggest deliberately poor modelling practice.

System dynamics models frequently contain multiple (and non-linear) feedbacks which readily elude our human cognitive capability—where multiple feedbacks exist, we need to be

able to develop comprehensive tests to assure that our models behave as they should. The problem that this presents to modellers is that, if we do not have the cognitive capability to understand the feedback mechanisms, how can we know that the tests we design and implement actually verify that our models work as they should?

We can improve our system dynamics modelling by use of molecules of system dynamics structure which we study in detail. Knowledge of these molecules and their behaviour can augment strategies for model testing, if we acknowledge that they can be constructed and tested separately then progressively combined to produce a model whose behaviour is compared with the model we have constructed through the top-down approach described.

Without systems engineering and the rigour it brings through progressive verification, we cannot expect to build sophisticated models of complex problems and have those models work properly.

When we combine systems thinking and system dynamics modelling with systems engineering concepts our focus changes to the fundamental building blocks of structure where, through a top-down approach, we define and build components parts (modules or building blocks), each designed with specific functionality in mind. We also have to pay close attention to management of the interfaces between the component parts we can build correctly functioning models. Each of these is developed for a specific purpose with specific representations of particular real-world problems.

We must also determine whether the model is a sufficient representation for our purposes. That is, we must test the model, ensuring as far as possible that it is correctly constructed and behaves correctly. We must also test that it faithfully represents the real world (Forrester, 1961: 115-129), but in both necessary and sufficient ways (Ashby, 1956: 202-218; McLucas, 2005: 151-152; Williams, 2002: 43; Wittenberg, 1992: 22-23).

Systems engineering as it is applied to system dynamics modelling can be described as a sequential process of requirements creation and integration into a model following the arrows in Figure 4, (adapted after Forsberg, et al., 2000: 116). The ‘Vee’ model is also described in standard systems engineering texts (Blanchard and Fabrycky, 1998; Sage and Rouse, 1999). The development of the system dynamics model follows the sequence from the top left of the ‘Vee’, to the bottom, then back up to the top right.

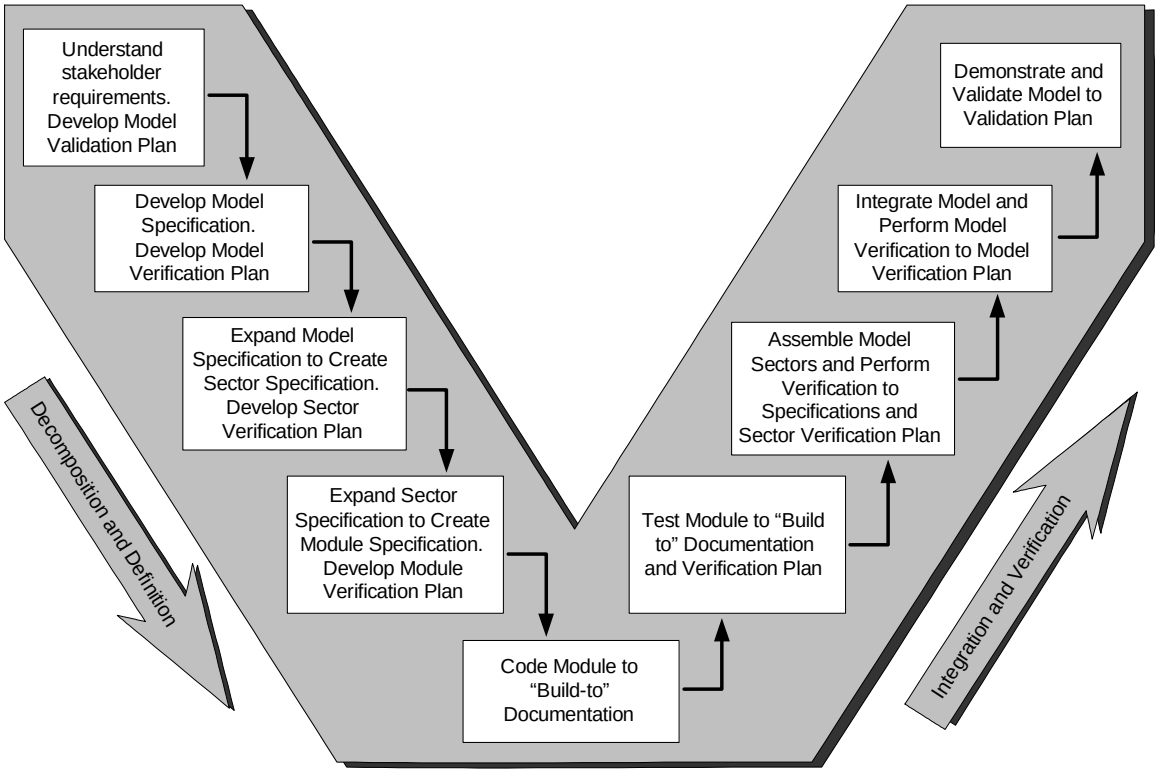

Figure 4. The basic systems engineering 'Vee' model applied to system dynamics model building.

Two essential model-building activities are *verification* and *validation*. Unfortunately these two terms are often used interchangeably, which leads to confusion. Verification and validation involve two distinctly different types of activities but which are inseparable when it comes to system dynamics modelling.

For our purposes, *verification* can be defined as (Jones, 1996: 94-96; Rakitin, 1997: 51-66):

> The process of determining whether or not the products of a given phase of the system dynamics modelling development cycle fulfil the requirements established during the previous phase.

Another way to view verification activities is that verification helps us answer the question: "Are we building the model *right*?'

In system dynamics modelling, *verification* is all about ensuring that the *governing business rules* have been correctly coded, that the *structure* in which those rules operate results in correct *replication* of the *reference modes of behaviour* identified in an earlier stage (and specified as *requirements* for the model we are building).

In essence, IEEE (1983) explains that *validation* is the process of evaluating models at the end of the model building-process to ensure that they comply with model requirements (from the client's perspective). When we validate a model we seek to answer the question: "have we built the *right* model?"

3.2.4.1. Verification—Considerations for Design of Testing

Verification involves designing and applying a sufficiently exhaustive set of tests that

measure how individual modules or complete models behave. This behaviour is compared with the modes of behaviour we have specified for individual modules or complete models.

Modules are designed to perform functions that are critical to the operation of the model as a whole. The tests we design and apply will establish the extent to which this is so.

Models must sufficiently and faithfully incorporate the governing business rules to produce behaviour over time that is representative of the reference modes defined a preceding stage. Again, the tests we design and apply will establish the extent to which this is so, including:

- *logical tests* to assure verification of parameters, integrity of dimensions of units, correct sequence of calculation, and correct form of output;
- *extreme-value tests* to assure stability under exposure to extreme conditions and extreme policies; and
- *mass-balance tests* to assure that physical flows do not violate the basic requirement for physical flows into a module, sector or model either accumulate or flow out.

3.2.4.2. Validation—Considerations for the Design of Testing

When we validate system dynamics models we seek to determine the extent to which two criteria are satisfied (Forrester, 1961: 115-129), the model must:

- generate behaviour that does not differ significantly from the real system, and
- explain real world behaviour through the structure and equations which reflect the real causal relationships in the real-world system.

The importance of the second criterion is that any number of models can be constructed to mimic the real world—that is, they can reproduce a given set of behaviours without faithfully representing real-world causal structures (cause-and-effect relationships).

This leads us to focus our validation activities for real-world problems on two types of test (McLucas, 2005):

- structural tests to assure boundary accuracy and structural accuracy; and
- behavioural tests to assure reproduction of behaviour, plausible behaviour prediction, identification of behavioural sensitivity, and identification of behavioural anomalies.

Validation testing, therefore, aims to identify cause-and-effect mechanisms and determine the extent to which the way we have represented them in our models is a sufficiently faithful representation of the real world to meet our needs. We must remain mindful of the fact that we cannot establish *truth* through system dynamics modelling (Sterman, 2000: 846). The best we can achieve is confidence that our models are necessary and sufficient representations of real-world cause-and-effect structures.

3.3. System Optimization and Balance

It should be noted that, as in engineering, the basic functions of system dynamics

modelling are taught and conducted bottom-up. That is, we take the basic structures and join them together to produce modules and thence sectors (subsystems) and models (systems). In engineering, therefore, there is a continual struggle during design to stay at an appropriate level of abstraction, or its inverse, aggregation, (principally by using the top-down approach) to ensure that requirements and complexity are appropriately managed.

The top-down approach has the additional benefit of contributing to system optimization and balance. That is, by using a higher-level methodology such as Soft Systems Modelling (SSM) before focusing on lower-level tools such as causal loop diagrams or influence diagrams, we can ensure that we stay focused on the problem in context and therefore achieve and maintain system-level optimization before dropping down to the detailed modelling tools (causal loop and influence diagrams) to apply rigour to the working elements of the model.

4. CONCLUSION

Systems engineering and system dynamics modelling have a common heritage. While they have diverged somewhat over the last 40 years, systems engineering has much to offer system dynamics modelling, particularly in terms of the discipline and rigour associated with requirements engineering, a top-down approach to managing and coping with complexity, validation and verification, and providing a mechanism for integrating a number of disciplines and specializations when engaging in group model building activities. Models built using the proffered methodology should enable model building with deliberate and careful management of the complexity introduced at each stage of the model building process. This should engender greater confidence that system dynamics models we build address clearly specified problems and that those models work correctly. As a consequence system dynamics modelling will be strengthened. The greatest potential gain accruing from application of additional rigour consistent with systems engineering practice will be improved acceptance of system dynamics modelling as a discipline, which would be of enormous benefit.

References

ANSI/EIA-632-1998, 1999, 'Processes for Engineering a System', Electronic Industries Association (EIA), Washington, D.C.

Ashby, W.R., 1956, 'An Introduction to Cybernetics', Chapman and Hall.

Blanchard, B.S. and Fabrycky, W.J., 1998, 'Systems Engineering and Analysis', 3rd ed., Prentice Hall.

Checkland, P.B., 1981 (1993), 'Systems Thinking: Systems Practice' John Wiley, Chichester, England.

Checkland, P.B. and Scholes, J., 1999, 'Soft Systems Methodology in Action' John Wiley & Sons, Chichester, England.

Coyle, R.G., 1996, 'System Dynamics Modelling: A Practical Approach', Chapman and Hall, London.

Defense Systems Management College (DSMC), 1990, 'Systems Engineering Management Guide', U.S. Government Printing Office: Washington, D.C.

Diehl, E. and Sterman, J.D., 1995, 'Effects of feedback complexity on dynamic decision making', In: *Organisational Behaviour and Human Decision Processes*, Vol. 62, No. 2.

Dörner, D., 1980, 'On the difficulties people have in dealing with complexity', in: *Simulation and Games*. 11: 87-106.

EIA/IS 632, 1994, 'Systems Engineering', Electronic Industries Association (EIA), Washington, D.C.
Faulconbridge, R., and Ryan, M., 2003, 'Managing Complex Technical Problems: A Systems Engineering Approach', Artech House, Boston, MA.
Forrester, J.W., 1961, 'Industrial Dynamics', Productivity Press, Portland, Oregon.
Forrester, J.W., 1971, 'Counter intuitive behaviour of social systems', in: *Technology Review* No. 73, January: 52-68.
Forrester, J.W., 1975, 'The impact of feedback control concepts on the management sciences', in: *Collected Papers of Jay W. Forrester*, Productivity Press: 45-60.
Forrester, J.W., 1985, 'The 'model' versus the modelling process', in: *System Dynamics Review*, Vol. 1, No. 1, Summer.
Forrester, J.W., 1994, 'System dynamics, systems thinking and soft OR', in: *System Dynamics Review*, Vol. 10, No. 2-3, (Summer-Fall): 245-256.
Forsberg, L., Mooz, H. and Cotterman, H., 2000, 'Visualising Project Management', 2nd ed., John Wiley & Sons.
Goodman, M.R., 1989, 'Study Notes in System Dynamics', Productivity Press, Portland, Oregon.
Graham, A., 1977, 'Principles on the Relationship Between Structure and Behaviour of Feedback Systems', Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA.
Hines, J., et al., (1996; 1997); 2000, 'Molecules of Structure: Building Blocks for System Dynamics Models', Version 1.4, LeapTec and Ventana Systems, Inc.
Homer, J.B., 1996, 'Why we iterate: Scientific modelling in theory and practice', in: *System Dynamics Review*, Vol. 12, No. 1: 1-19.
IEEE, 1983, 'ANSI / IEEE Standard 729-1983: IEEE Standard Glossary for Software Engineering Terminology'.
IEEE-STD-1220-1994, 1995, 'IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process', IEEE Computer Society, New York, N.J.
Kleinmuntz, D.N., 1985, 'Cognitive heuristics and feedback in dynamics decision environment', in: *Management Science*, Vol. 31, No. 6: 680-702.
Kleinmuntz, D.N., 1993, 'Information processing and misperceptions of the implications of feedback on dynamic decision making', in: *System Dynamics Review*, Vol. 9, No. 3 (Fall 1993): 223-237.
Kline, S.J., 1995, 'Conceptual Foundations for Multidisciplinary Thinking', Stanford University Press, Stanford, California.
Jones, C., 1996, 'Software defect-removal efficiency', in: IEE Computer, Vol. 29, No. 4, April: 95-96.
Lake, J., 1996, 'Unraveling the Systems Engineering Lexicon', Proceedings of the INCOSE Symposium.
McLucas, A.C., 2005, 'System Dynamics Applications: A Modular Approach to Modelling Complex World Behaviour', Argos Press, Canberra, Australia.
McLucas, A.C. and Ryan, M.J., 2005 (forthcoming), 'Combining Generic Structures and Systems Engineering to Manage Complexity in System Dynamics Modelling' in: *Proceedings of International System Dynamics Conference*, System Dynamics Society, Boston, MA.
Morecroft, J.D.W., 1992, 'Executive knowledge, models and learning', in: *European Journal of Operations Research*, Vol. 59, No. 1: 9-27.
Morecroft, J.D.W. and Sterman, J.D. (eds) 1992, *European Journal of Operations Research – Special issue on 'Modeling for Learning*.

Morecroft, J.D.W. and Sterman, J.D., 1994, 'Modelling for Learning Organizations', Productivity Press, Portland, Oregon.
Mosekilde, E. and Larsen, E.R., 1988, 'Deterministic chaos in the beer production-distribution model', in: *System Dynamics Review,* Vol. 4, Nos. 1-2: 131-147.
Mosekilde, E, Larsen, E.R. and Sterman, J., 1990, 'Coping With Complexity: Deterministic Chaos in Human Decision Making Behaviour', in: J. Casti and A. Karlqvist (eds.), *Beyond Belief: Randomness, Prediction and Exploration in Science*. CRC Press, Boston, 1990.
Paich, M. and Sterman, J.D., 1993, 'Boom, bust, and failures to learn in experimental markets', in *Management Science*, Vol. 39, No.12: 1439-1458.
Rakitin, S.R., 1997, 'Software Verification and Validation: A Practitioner's Guide', Artech House: 51-66.
Richardson, G.P., 1990, 'Feedback Thought in Social Science and Systems Theory', University of Pennsylvania Press.
Richardson, G.P. and Pugh, A.L.III, 1981, 'Introduction to System Dynamics Modelling', MIT Press / Wright-Allen, Portland, Oregon..
Sage, A.P. and Rouse, W.B., 1999, 'Handbook of Systems Engineering and Management', John Wiley & Sons, New York.
SECMM-95-01, 1995, 'Systems Engineering Capability Maturity Model', Version 1.1, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, P.A.
Sterman, J.D., 1989a, 'Misconceptions of feedback in dynamic decision making', in *Organisational and Human Decision Processes*, No. 43: 301-335.
Sterman, J.D., 1989b, 'Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making Experiment', in: *Management Science,* Vol. 35, No. 3: 321-339.
Sterman, J.D., 1989c, 'Misperceptions of feedback in dynamic decision making', in: Milling, P.M. and Zahn E.O.K. (eds), *International System Dynamics Conference: Computer-Based Management of Complex Systems*. International System Dynamics Society, Stuttgart: 21-31.
Sterman, J.D., 1994, 'Learning in and about complex systems', in: *System Dynamics Review,* Vol. 10, No. 2-3, (Summer-Fall): 291-330.
Sterman, J.D., 2000, 'Business Dynamics: Systems Thinking and Modelling for a Complex World', Irwin McGraw-Hill.
Sterman, J.D., 2002, 'All models are wrong: reflections on becoming a systems scientist', in: System Dynamics Review, Vol. 18, No. 4, (Winter): 501-531.
Stevens, R., Brook, P., Jackson, K. and Arnold, S., 1998, 'Systems Engineering: Coping With Complexity', Prentice Hall, London.
Sweeney, L.B. and Sterman J.D., 2000, 'Bathtub dynamics: initial results of a systems thinking inventory', in: *System Dynamics Review,* Vol. 16, No. 4, (Winter) 2000.
Wiener, N., 1948, 'Cybernetics: or Control and Communication in the Animal and Machine', MIT Press, Cambridge, MA.
Williams, T., 2002, 'Modelling Complex Projects', Wiley
Wittenberg, J., 1992, 'The idea of a model in Kuhnian Science', in: *System Dynamics Review*, Vol. 8, No. 1, Winter: 21-33.
Wolstenholme, E.F., 1990, 'System Enquiry: A System Dynamics Approach', John Wiley and Sons, Chichester. UK.