System Dynamics: a Core Systems Engineering Capability

Kim Warren

Systems Engineering (SE) is moving from a document-based discipline to model-based systems engineering (MBSE). SE is also reaching out from its historic focus on physical systems to tackle wider issues in enterprise planning and management. Time-based simulation with System Dynamics (SD) is an ideal support for both these trends and is therefore a skill that every systems engineer should possess. After a brief explanation of each field, a simple 'Living Business Model' shows how SD models work and their benefits. We then outline the wider scope of applications for SD modeling and suggest first steps to start acquiring SD skills.

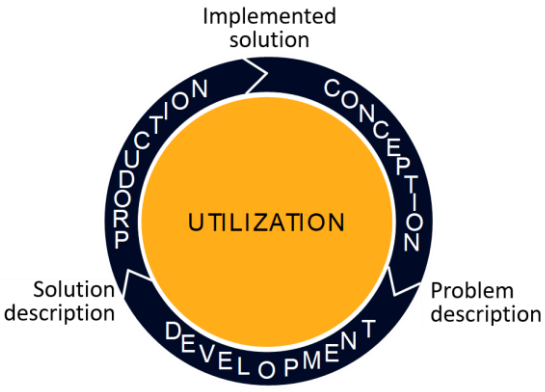
(This paper is also available in three parts on LinkedIn at sdl.re/LIPSESD1, sdl.re/LIPSESD2 and sdl.re/LIPSESD3).

What is Systems Engineering (SE)? … for non-systems-engineers[1]

The SE professional body, INCOSE (incose.org) defines "a system" as *... a construct of different elements that together produce results not obtainable by the elements alone*. Those elements can include people, hardware, software, facilities, policies, documents; everything required to produce system-level behavior and performance.

"Systems Engineering", then, is *... an engineering discipline [for] creating and executing an interdisciplinary process to ensure that customer and other stakeholder's needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system's entire life cycle* – a definition that reflects SE's historic focus on physical and technical systems and projects. SE has in recent decades extended its scope and is deployed for a wide range of organisational issues, and even for social, economic and environmental challenges. SE can also help 'engineer' and manage the ***continuing operations*** of enterprises, both the whole enterprise, and key functions or activities.

Figure 1: A system's Acquisition and Utilization



A system life-cycle may be viewed as two main phases: (1) ***acquisition*** in which the system is conceived, developed and created, and (2) ***utilization*** – where the system is operated to fulfil its purpose and produce desired outcomes (Figure 1). Later "transformation cycles" may re-conceive and modify (re-engineer) the system, often to fulfil extended purposes.

These phases are clearly evident in the life-cycle of an enterprise, and also apply to initiatives that an enterprise may undertake. The Acquisition/Utilization split may shift, however, depending on the level at which the system is considered. Opening a retail store, for example, involves acquisition, as it is planned and built, and utilization as the now-opened store is operated for many years. However, the growth and management of a whole chain of stores is a continuing operation, within which the many store-openings are merely a stream of events.

Finally, ***retirement*** ceases the system's operation when no longer viable or needed and disposing of its parts. But retirement rarely features in enterprise management – few executives or investors plan for going out of business, or for 'retiring' change-initiatives that they undertake.

[1] I am grateful to George Sousa for his guidance on the essence of systems engineering.

Model-based Systems Engineering (MBSE) The SE body-of-knowledge or SEBoK (sebokwiki.org) explains that SE is moving from documentation-led approaches, which "*suffer a lack of precision, inconsistencies between elements, and difficulties in maintaining and reusing the information*", to a more model-based discipline (SEBoK p.39). MBSE is "*the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing through development and later life cycle phases*".

So far, it seems, MBSE is still applied only in pockets within organizations and unevenly across industry sectors. Further evolution of the modeling tools is called for, plus a workforce skilled applying MBSE. Software tools employing The Systems Modeling Language (SysML) enable SEs to formalise many types of model, mostly as linked-block diagrams. But such models are beyond the scope of this article, which focuses on ***quantified simulation models that generate time-based analysis, and projections of system-wide behaviour***.

What is System Dynamics (SD)?

The International System Dynamics Society (systemdynamics.org) defines SD as *"... a computer-aided approach to policy analysis and design. It applies to dynamic [time-based] problems in complex social, managerial, economic, or ecological systems*." So SD ***simulates*** how any physical, social, economic, and/or environmental system behaves over time, or indeed a system featuring elements from all these domains. Here we focus on modeling enterprises (a commercial business, public service or non-profit organisation) or any initiative it undertakes.

SD is one of three related dynamic modeling methods. ***Discrete-event simulation*** (DES) is very widely used to model how entities move through a process, such as queuing cases, production systems or supply-chains. ***Agent-based modelling*** (ABM) simulates the actions and interactions of autonomous agents, to assess their effects on a wider system. It is especially powerful with geo-spatial phenomena such as the spread of a disease. SD, in contrast, is a ***continuous*** simulation method that captures interactions between ***populations*** of things, people, cash or materials, including intangible factors, such as states-of-mind, information and certain quality measures. (*See more on these methods in Maidstone, 2012*). These features of SD modeling lead to two key benefits.

- First, the focus on aggregate populations makes SD models ***compact and quick to build***. But this has a cost – SD models say nothing about individual entities. So, for example, service quality may be OK on average, but lousy for a specific customer. SD can model sub-groups, but this is still not the entity-level detail of DES or ABM. It can be useful to combine these tools in hybrid models (*Mustafee et al, 2015*).
- Secondly, SD's capture of all physical and non-physical entities in a consistent and rigorous manner enables models to simulate ***all aspects*** of a project, initiative or enterprise – physical, human, financial.

A sound SD model not only matches a system's observed or likely ***performance*** – many techniques can fit trends and create forecasts – but should also match everything ***driving*** those outcomes. Like a good working model in any field, then, it mimics all the behaviour of the thing it portrays, albeit at an aggregate level. In the extreme, an SD model can be a high-level "digital twin" of the enterprise or issue, which can be used to help manage that system by continual updating with latest real-world information.

This aim may seem over-reaching – surely we cannot capture and simulate the complexities of how a whole enterprise functions? However, a logical, step-wise approach does make the aim achievable, and is not even especially difficult. The resulting Living Business Models capture, intuitively and quantitatively, mechanisms that spreadsheet-based efforts at business modeling cannot handle: long-term ***accumulation*** processes (e.g. winning and losing customers or staff), including ***intangible*** factors (e.g. data, staff motivation, reliability), ***interdependencies*** between those accumulating factors, which often include ***delays***, ***thresholds*** that trigger

discontinuous change. Critically, the interdependencies create ***feedback*** mechanisms that can accelerate or stifle growth or decline, or cause boom-and-bust episodes or cyclicality.

Dynamic modeling is different from process modeling!

Many SEs are very familiar with process modeling or mapping, but are less aware of dynamic modeling, so it is important to understand the fundamental difference between the two. SD, DES and ABM all share the feature that they represent the "***things and materials***" that exist and interact, and that are created, leave or move through a system. Elements in a dynamic model are therefore exclusively ***nouns***, such as people, customers, products, units, cash, data and so on. These methods also ***quantify*** those things and materials in the system and simulate how those quantities change over time.

This is quite different from ***process modelling*** approaches that map, qualitatively, the connected processes that ***act on*** those things and materials. Process model elements are therefore mostly ***verbs*** – hire people, develop products, receive cash, and so on. (*See Appendix 1 or sdl.re/LIPprocessVdynamicModeling for more*)

Elements of an SD model are:

> … "accumulating stocks" or just ***Stocks*** – quantities of things or materials that exist and that define the state and rate-of-change of a system at each point in time, such as inventory, cash, staff, customers, machines. These are fundamental to how the real world works. (*See "What is an Asset-Stock and Why Should You Care?" at sdl.re/LIPstock*.)
>
> … the "flow-rates" or simply ***Flows*** that cause Stocks to grow or decline, such as orders/day, cash-flow $/week, people/month;
>
> … all ***other items*** – including constant parameters such as hours per working week, ratios like demand-supply balance, variables such as service quality, and performance results.

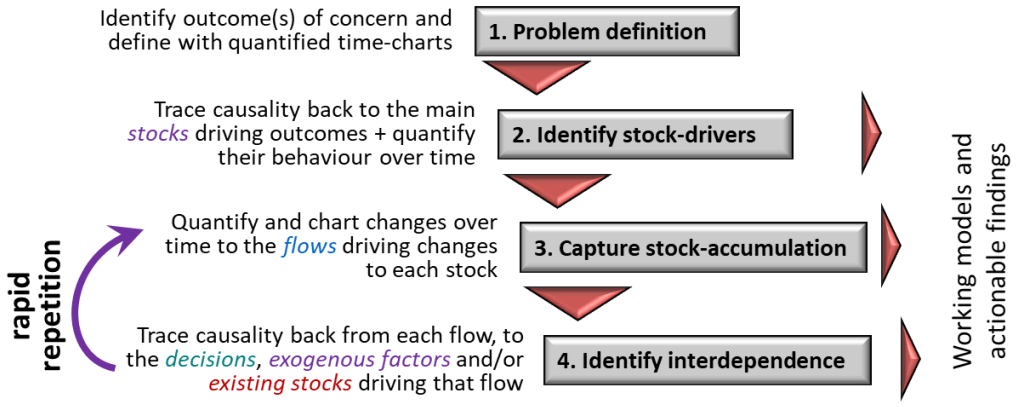
An SD model also includes the ***causal relationships*** between any items that depend on each other. Most are simply arithmetical; for example, *sales/month* = *customers* * *sales/month-per-customer*; or *work capacity* (*person-hours/day*) = *staff* * *hours/day-per-person*. Other causal relationships are less readily formulated but can be "looked up"; for example, how *staff turnover* changes with rising work pressure, or how *sales/month-per-customer* change with changing prices. This allows models to capture important threshold effects - for example, *sales per customer* may be near-zero at any too-high *price* but escalate sharply as *price* falls to an acceptable threshold; or *staff turnover* may rise sharply when work pressure reaches a threshold that staff will tolerate.

Building SD models Reliable SD models are most easily built by a rigorous abductive process (Figure 2), explained at sdl.re/LIPagileSD and the paper at sdl.re/agileSD. This process starts from performance outcomes, and works back along chains of causality, validating each causal relationship in turn. This process quickly reveals that …

- ***performance outcomes*** depend on one or more stocks – sales depend on the customers-stock; service capacity depends on the service-staff-stock, for example
- changes to all such stocks are driven exclusively by the ***flow-rates*** of quantities that are added to, or lost from those stocks – the number of customers or staff today, for example, is precisely by the number last month, plus and minus any customers or staff won or lost during the month
- those flow-rates depend on ***current stocks*** – the flow of *customers won/month* depends on the stock of *sales people*, and *staff turnover* depends on the workload driven by the stock of *customers.* This is the mechanism that explains the interdependencies in the system.
- one or more flow-rates depend on the current quantity of ***its own stock*** – *customers won/month* depends on the current stock of *customers* through a word-of-mouth process, and *staff turnover* depends on the

work pressure that reflects the current stock of *staff*. This is the mechanism that causes models to capture real-world feedback processes.

Figure 2: The agile process for developing SD models



There are two other common approaches to building SD models. First, we can ***start from well-established generic models*** that have been proven to match whole classes of system. Many similar systems share common structures, often with common implications for design and management of those systems. Consultants who focus on a specific application domain are highly skilled in wielding such generic models, adapting them to any specific case.

Generic templates exist for many small issues – for example, how service quality reflects service demand and service staff capacity; how marketing and sales effort moves customers through stages of awareness and loyalty; how maintenance affects the deterioration of physical assets and the resulting system reliability. There are also much larger templates – common structures have been found for whole classes of business-types (retailing, professional-service firms, media businesses, and so on), and generic models exist for different kinds of environmental challenges, large-scale projects, social-welfare issues, and so on.

Another style of model-building starts by developing qualitative causal diagrams. Widespread consultation captures how stakeholders believe the interdependencies in a system to work. This results in what is known ***as a shared mental model*** in the form of a causal-loop diagram (CLD) of relationships and feedback loops throughout the system. From this qualitative map, modelers identify the accumulating stocks, seek data, then construct and validate a working quantitative model. A good explanation of this approach can be found in Vennix, 1996, and for more technical guidance on building working models, see Sterman, 2000.

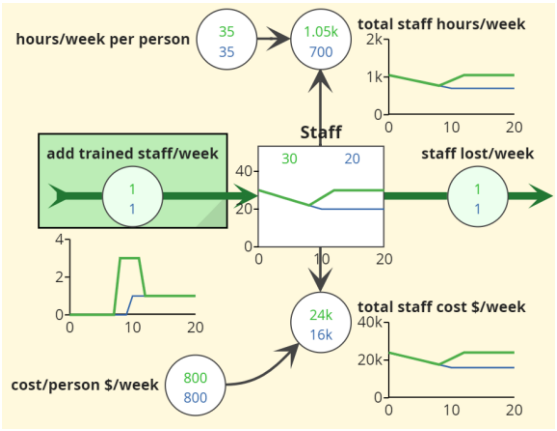
Many practitioners go no further than to seek insights from the qualitative CLD in what is known as 'feedback systems thinking'. There is a widespread belief that such qualitative feedback diagrams are SD, but the focus in this paper is on the potential contribution of ***quantified SD simulation models.***

A simple SD project example

How SD models function is best explained with a worked example. A team of staff is working to complete a project over 20 weeks. The project consists of a set of tasks, each requiring some person-weeks of effort to complete. Staff may leave but can be replaced by others who are hired and trained.

Figure 3 shows a small model-section in which the number of staff on the project are lost and replaced over 20 weeks. Those staff drive both work capacity and cost. The figure compares a policy that reacts late to staff losses, and then only replaces those currently leaving (blue) with a policy that reacts earlier *and* replaces staff previously lost (green). Try this model for yourself at sdl.re/SEstaff.

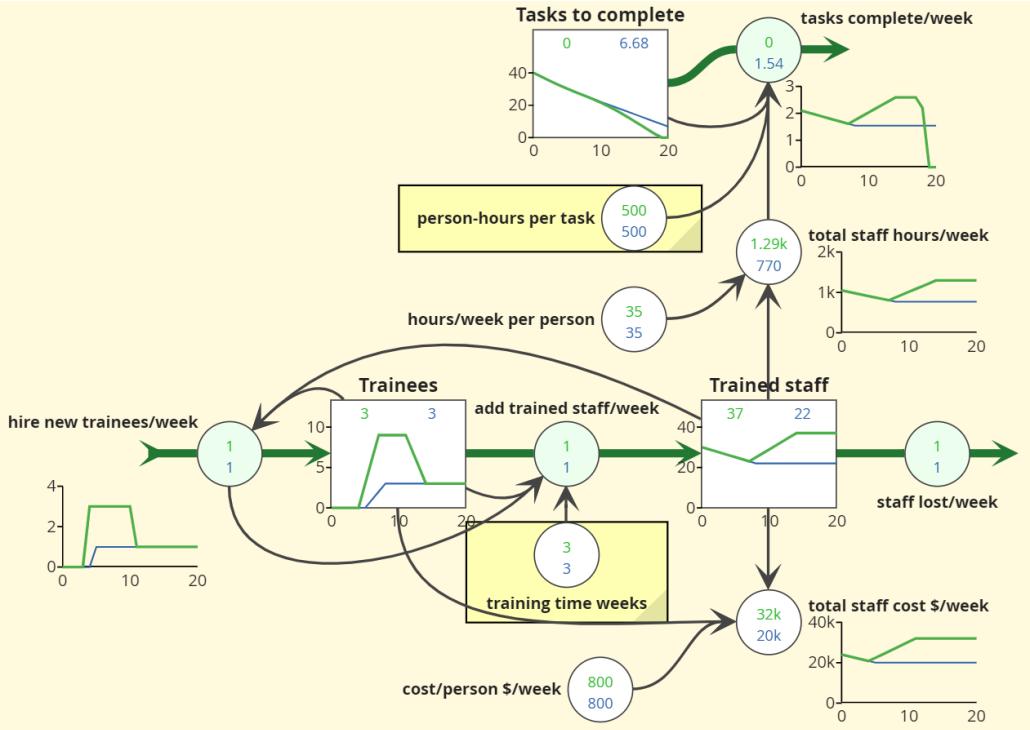
Figure 3: How a Stock of staff changes over time



These models are not complicated! – Think of this image as a set of linked charts on top of a spreadsheet. Each item is a spreadsheet column with its name in the top cell and values for each week in the cells below. The arrows are like cell-references, showing what is calculated from what – so you can't make cell-reference errors! The box for Staff and the thick arrows for staff added and lost simply highlight the unique mathematical relationship between a Stock and its Flows. Each item holds the formula for calculating its values from those it depends on – and that formula is in natural language, not Excel-code!

In figure 4, only *trained* staff are working through the Stock of tasks needed to complete the project. In the blue scenario, lost staff are replaced if there are fewer than 25 people in total. But new hires need 3 weeks' training, so we have too few staff and the tasks are not done fast enough to complete the project in 20 weeks. In the green scenario, we react earlier to the staff shortage, hire staff faster, *and* continue hiring up to a larger total number. This is just enough to complete the project within 20 weeks, although of course at a higher cost. Explore the model at sdl.re/SEproject, trying to complete the model on time at minimum cost.

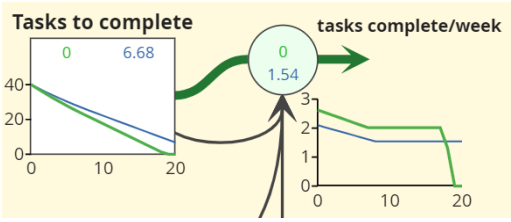
Figure 4: How changing staff numbers drive completion of a project.



Testing process improvements Appendix 1, comparing process-v-dynamic modeling, explains that key processes are often located at the flow-rates in SD models and determine how fast those flows run. Key processes here (each made up of more detailed activities we do not show) are hiring trainees, training them, and completing project tasks.

We can use the model to test the benefit of making process improvements. Figure 5 shows the result with training time cut to 2 weeks and person-hours per task cut from 500 to 400. The project completes by week 20, even if staff numbers decline and are not re-built. You will also see in the model that the shorter training time delays the need to hire by 1 week and reduces costs by cutting the number of unproductive trainees.

Figure 5: Project completion with improved processes for training and task completion



This project-completion demo is of course very simplified – SD models can add the many complexities needed to model real projects, and indeed SD has long been used for exactly this purpose (Lyneis and Ford, 2007). See also information at systemdynamics.org/fluor on how Fluor Corporation has used dynamic project models.

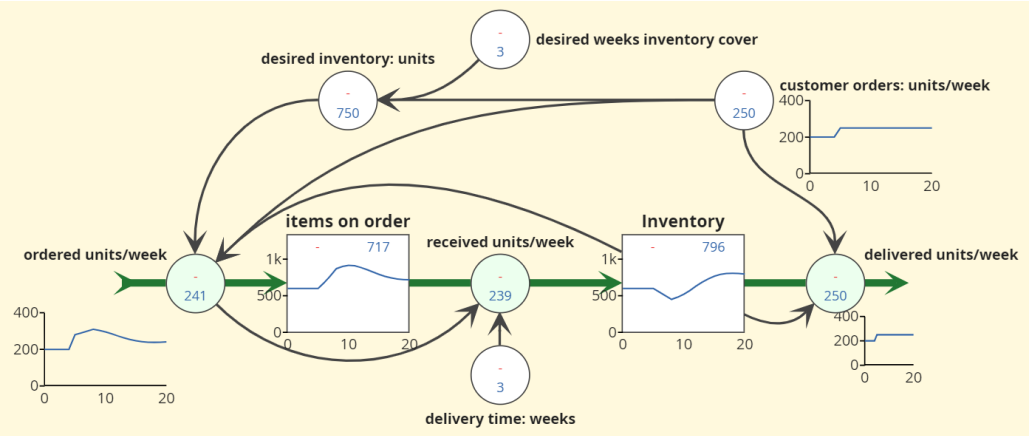
SD modeling of physical systems

Although project management SD models are mostly built around stocks of work-to-do, rather than on the physical entities to which that work is done, models may also simulate changes to quantities of relevant physical elements – units produced, tons of material consumed and so on. Late delivery of raw materials, for example, may delay work completion, even if adequate staff are available, and consumption of materials drives costs.

The product development domain features a similar focus on work-to-be-done. SD models of product development track the completion of features needed for a product to be deemed adequate by customers. Such models may go on to assess how the addition of further features drive growth in adoption and purchase by those customers. Setting this alongside a similar structure for *rivals'* product development dynamics enables more powerful models of competition to be built than can be achieved by considering prices and marketing alone.

Since SD's foundations arise from principles of engineering control theory (Forrester, 1968), it has from the start been applied to a wide variety of physical-system challenges. ***Supply chains*** are of course made up of interconnected stocks of items and materials, between which goods and materials flow. SD models capture the high-level dynamics of supply chains – how aggregate quantities rather than individual entities move and change over time – but can add links to non-physical factors, such as workloads and the financial value of those materials. Figure 6 shows a business holding inventory in order to meet customer-orders and replenishing that inventory with orders placed on a supplier and received after a delivery delay. The figure shows how a certain policy for setting orders to the supplier changes the actual order-rate and inventory when customer orders increase.

Figure 6: Changing orders and inventory levels in a simple supply chain



The model (*at sdl.re/SEsupplychain*) demonstrates the complexities involved in designing an ordering policy that best-meets changing customer orders while minimising the costs incurred by holding inventory (Sterman, 2000: chapter 17). Models can also account for often-powerful intangible factors and their impact on policy for managing supply chains. Fear of stock-outs, for example, may cause over-ordering, with potentially damaging and costly implications for inventory levels and flows.

Asset management is another domain in which SD models of physical factors have been applied. Such models can track populations of different types of equipment through a typical life-cycle – after bedding-in, units have a long reliable life, before degenerating and becoming quite unreliable. The commitment of staffing and expenditure to the maintenance, refurbishment and replacement of assets is a complex challenge that must balance system-performance aims, notably reliability, against the considerable costs of sustaining the network of physical assets. (*See a demonstration model of such a challenge at sdl.re/assetpipeline*).

Physical factors feature in many SD models of environmental challenges, such as the management of water resources, natural resources (crops, fish, livestock …), and climate-change impacts (Ford, 2011). Again, the special contribution of such models is the ability to capture interactions between physical and non-physical factors in a faithful simulation of an entire situation or episode. Take the case of a large-scale engineering solution to rejuvenate water quality and wild-life in a moribund lake, explained at sdl.re/LakeModel[2]. The project required integrated modeling of the hydraulics and water quality, power-generation, the physical engineering, and the financial business case. Achieving this integrated simulation depended on capturing the knowledge of experts from several disciplines in a shared mental model, enabling all parties to see the relationships between their own part of the system and the whole. The resulting model enabled all parties to see, clearly and immediately, the impact of alternative assumptions and options for the project.

Modeling a new business initiative

Turning attention back to models for business cases, the model at sdl.re/SEnewservice extends the scope of the small project model in Figure 4 by making it the first phase of a larger initiative:

- The wider initiative concerns developing a new service that a business believes can be sold to clients (mostly customers for existing products, not included in the model).
- Once the new service is completed, project staff are redeployed onto other tasks, but at the same time sales people are deployed to capture clients for that new service.
- Clients require support from service staff and leave if service quality is poor.
- Clients pay us fees for the service, generating rising revenue.
- However, the business incurs project-staff costs during many weeks before the service can be sold, and also incurs sales-effort costs before revenue starts to rise.
- The net difference between the revenue generated and costs incurred is the net cash flow of the initiative. Cumulative costs drive a heavy total investment in the initiative before rising revenue starts to repay that investment.

[2] This case is courtesy of Copernicos Groep.

A screen-shot of the full model (at sdl.re/SEnewservice) is in Figure 7. This model is, in effect, the 'design' of the new initiative, and includes many assumptions about how it is expected to work (the highlighted parameters) – how much work is needed for each staff activity, how much people cost and how much revenue clients generate. More subtle mechanisms include the inefficiencies that arise from deploying too many staff on the development project, the increasing effort needed to win later clients, and the impact of service quality on client losses.

This is an example of a complete model of a strategic issue for the enterprise, and includes most of the key features of SD noted earlier:

- The focus is on how the system ***performs***, quantitatively, over time.
- The ***stocks*** (staff and clients), all of which are changed by ***flow-rates***, drive all performance outcomes
- … and the training pipelines introduce ***delays***.
- All causal relationships are captured with ***simple arithmetic*** – except for the discontinuous, ***threshold*** relationship between service adequacy on client losses, which is 'looked up'
- All flow-rates are driven by one or more stocks (plus our decisions), which capture ***interdependence*** between system elements
- … and ***feedback*** arises because some flow-rates depend on the level of their own stocks (for example, the client loss-rate depends on the balance between the current number of clients itself and the number of service-staff).

Multiple uses for Living Business Models

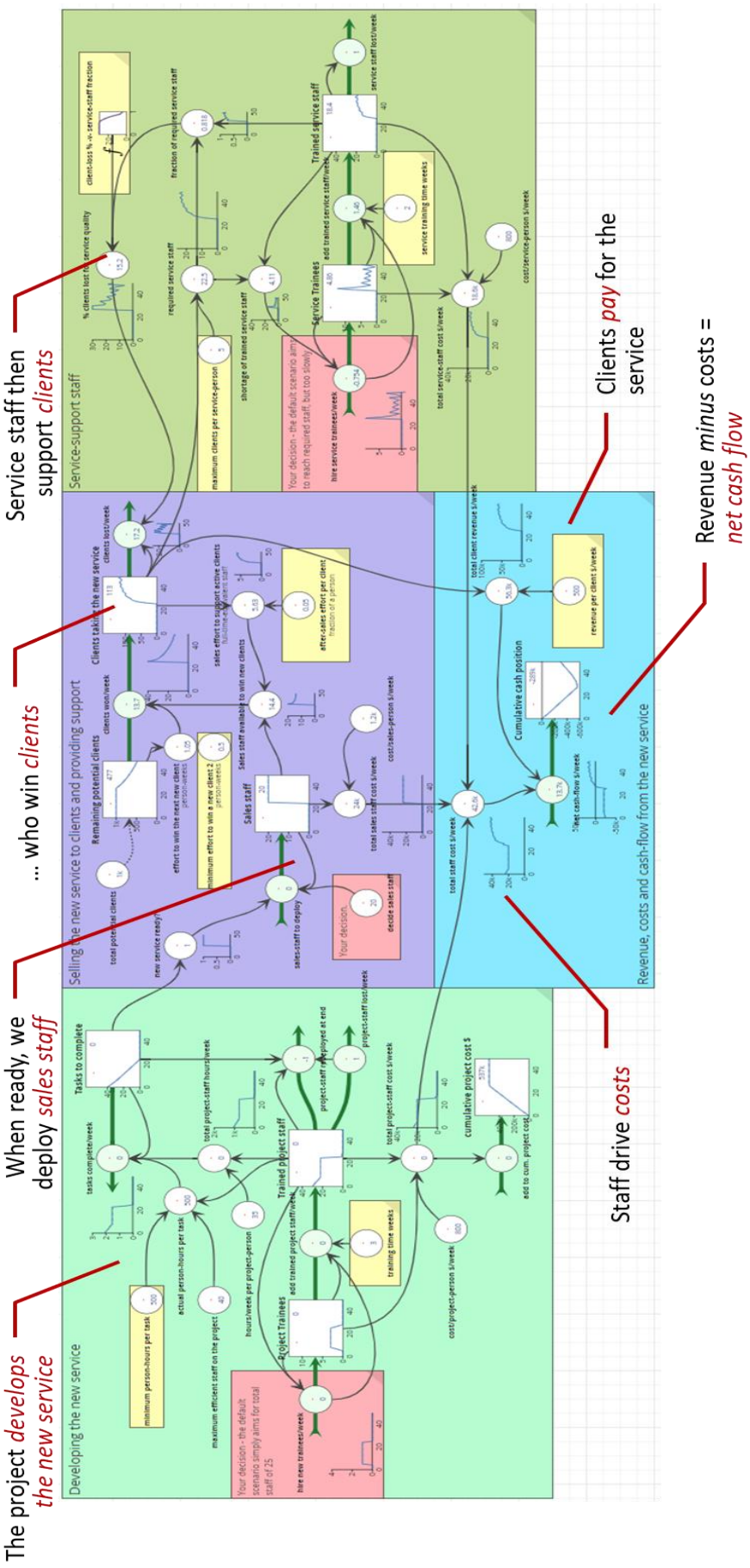
SD models may be deployed at multiple levels in an organisation, and for diverse purposes (Figure 8). Most SD applications in business cases have concerned understanding and addressing ***specific issues***, whether small-scale and quick, such as fixing a service quality problem, or larger-scale such as acquiring and merging with another company. However, it is increasingly apparent that models can be used for ***on-going*** planning and management of continuing operations. The model in Figure 7, for example, could be routinely updated with actual progress on the new-service initiative and so be of continuing value for managing that initiative. Such continuing use may, of course, lead to the model being modified as more is learned about how the real world system is working.

Figure 8: Examples of the multiple purposes and levels for using SD models

Level	Purpose: … for a continuing *plan*	Purpose: … for a specific *issue*
… for the *whole* organisation	The *strategic plan* for a whole company. The strategy for a *new enterprise*.	Defeating a *competitor*. *Merging* with another company. Responding to a *recession*.
… for a *team*, *function* or *department*	The *marketing* plan. A *human resources* strategy. The *IT strategy*.	Fixing poor *service quality*. Changing a team's *skills*. Cutting new product *lead-times*.

(*There is a still-lower level at which SD models can be used – for shorter-timescale operational processes, such as the hiring example in Appendix 1.*)

Figure 7: A model on developing, selling and supporting a new client-service



The following demonstration models illustrate the variety of purposes for which SD-based Living Business Models can be deployed:

- The start-up model for ***a neighbourhood restaurant***, at sdl.re/BMrestaurant4e is a complete enterprise, albeit a simple one. This model demonstrates the standard classes of Stocks or "resources" found in most business cases; customers, staff, products, capacity (seats!) and cash. It is also an example of a generic business-model template – every neighbourhood restaurant in the world conforms with this model, although all parameters will of course be case-specific. A restaurant ***business*** is simply the sum of many such single-unit models, plus structures to model central business functions.
- Many organisations face threats from ***competition***. The model at sdl.re/BMpharma4a, simulates an intensely competitive episode in a pharmaceuticals market. A new competitor product aims to steal the physicians who currently take our established, market-leading product, so how to set pricing to fight them off while protecting profitability as much as possible?
- The model at sdl.re/BMstaff6b forms the basis for ***a continuing functional plan*** – staff development. It captures not only the changes to staff numbers as people are hired, promoted and leave, but also the consequences for staff-experience and salary costs at each level.
- The model at sdl.re/BMITsupport4a concerns a service-quality crisis at a small IT-support company. The model was used to figure out how to recover from a crisis that had built up over many months but can also be used to help avoid such a crisis in the first place.

For more on the power of SD-based Living Business Models, see sdl.re/PowerOfModels and sdl.re/LIPbusinessmodels.

Conclusions, and first steps for acquiring SD-modeling skills

This article has explained what SD modeling is and demonstrated some of its benefits. SD models can, of course, be much more extensive and may be subjected to highly sophisticated exploration of scenarios and sensitivities. How might such models contribute to SE practice?

- Project management simulations are already widely used to help plan and manage the acquisition of physical systems, and the principles are readily transferable to ***planning and managing*** the creation of new enterprises and the undertaking of any initiative for existing enterprises.
- In the Acquisition phase, simulation can confirm that any new enterprise or initiative is ***capable*** of fulfilling its purpose, before real financial and human resources are committed to creating it.
- In the Utilization phase, when the new enterprise or activity is up and running, simulation can help ensure that it ***does*** fulfil its purpose and continues to do so to a sustained high level of performance – just as the control-panel in a power plant or chemical plant helps operators manage these physical systems.
- Modeling can also be used to ***fix problems*** that arise as the enterprise or initiative continues to progress.

Although modeling the most complex cases is a demanding challenge, needing much experience, any reasonably numerate individual can acquire the basic skills in a short time. ***If you can build simple spreadsheet models, then you can build simple SD models!***

> For the basic technical skill, see free online classes at sdl.re/modeling-getting-started/ - about 2.5 hours of video explanation and "follow-me" demonstrations.
>
> Next, see the short guide to the AgileSD process for developing models with users or clients at sdl.re/agileSD – the term "agile" reflecting the features of the process that deliver working,

quantified simulations from the start, including continuous validation with real-world data, with the full involvement of the users.

An outline of the most common structures that feature in business cases is in *Strategy Dynamics Essentials* (Warren, 2016), see sdl.re/essentials.

Further instructional material is at sdl.re/courses, and more info at youtube.com/strategydynamics.

Comments on this article can be posted to the Strategy Dynamics Network at linkedin.com/groups/1688847. Questions on the SD modeling method and our learning resources can be posted to our Forum sdl.re/forum.

References

Ford, A., 2011. System dynamics models of environment, energy and climate change. In *Extreme Environmental Events*. Springer, New York, NY.

Forrester, J.W., 1968. Industrial dynamics—after the first decade. *Management Science*, *14*(7), pp.398-415.

Lyneis, J.M. and Ford, D.N., 2007. System dynamics applied to project management: a survey, assessment, and directions for future research. *System Dynamics Review*, **23**, pp. 157-189.

Maidstone, R., 2012. Discrete event simulation, system dynamics and agent-based simulation: Discussion and comparison. *System*, **1**, pp. 1-6.

Mustafee, N., Powell, J., Brailsford, S.C., Diallo, S., Padilla, J. and Tolk, A., 2015, *Hybrid simulation studies and hybrid simulation systems: definitions, challenges, and benefits*. In *Winter Simulation Conference*, 2015 (pp.1678-1692). IEEE.

Sterman, J., 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World.* McGraw-Hill, New York.

Vennix, J., 1996. *Group Model Building*. Wiley, NJ.

Warren, K., 2016. *Strategy Dynamics Essentials*. Strategy Dynamics Ltd.

Appendix 1: Dynamic Modeling -v- Process Modeling

The use of 'models' to support business strategy and management is a common idea, but the term "business modeling" means many different things to different people.

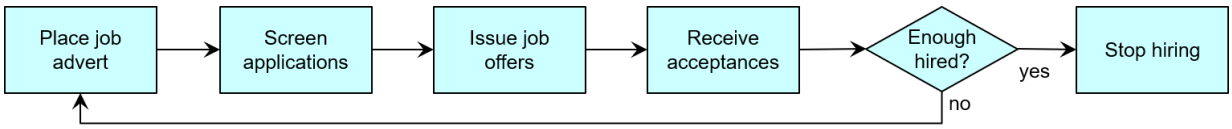
Note first that we are ***not*** talking here about "business models" in the sense of a high-level explanation of what a business chooses to do (what customers to serve, with what products and services), and how it makes money. The frameworks used for those models mostly lead to descriptive diagrams rather than models that actually ***do*** anything (see the short article on this topic at sdl.re/LIPBusinessModels).

Rather, we are talking about the many types of 'model' that executives and consultants may use to portray how a business – or a part of it – functions and performs. Such models are used to aid management in designing or managing the business or critical functions within it.

Process Modeling basics

Many people are familiar with process-mapping approaches to modeling the activities that together enable a business to function. Indeed, process mapping is a very widespread professional discipline, of fundamental importance in providing a platform for designing or redesigning business operations.

At its simplest, a process map or diagram lays out the sequential actions needed to accomplish a simple functional task, such as this basic hiring process …



Similar process diagrams may be laid out for "customer journeys" (how customers are taken on, or how sales or service activities happen), for supply-chains, for accounting processes and so on – that is, for all activities and functions of a business.

Process mapping is recursive, so for example this hiring activity may appear as a single element in a wider view of how an organisation's staff-management operates. Taken to its full extent, process mapping can provide an architecture for ***the whole enterprise*** – though of course this cannot be captured on a single picture, but requires many, related diagrams. Such "enterprise architectures" are widely used to design IT systems and services (see the Open Group explanation of EAs at opengroup.org/subjectareas/enterprise).

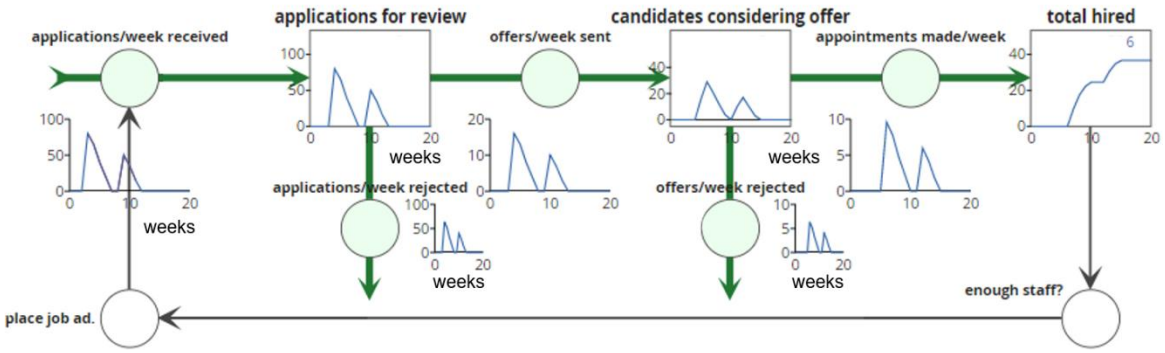
What exactly is process modeling doing? As the hiring example illustrates, process-modeling captures the ***activities*** that act on the items that make up the enterprise, such as materials, machines, people, customers, cash, and information. Most elements on a process diagram are therefore ***verbs*** (do A, then do B, then …). Although process-mapping is a qualitative discipline, it can and should be rigorous, and amenable to validation – it is easy, for example, to check that the steps in the hiring process above are in fact what happens to each job application. Any real-world hiring activity may, of course, involve different or additional processes, such interviewing.

Dynamic Modeling

Dynamic modeling is rather less well-known than process modeling, perhaps because it is often done by specialists, and because historically it has mostly been used for technical functions of the business, such as the

design of production facilities and supply-chains. Three main methods are used for dynamic modeling – discrete event simulation, agent-based modeling and system dynamics – but the feature they share is that they all quantify how ***numbers of items*** in a business or activity ***change over time***, how fast they ***move*** between different states, and how those changes and movements ***depend on*** other items in the system. For example, the number of job applications screened each week depends on how many applications have been received, and on how many HR staff are available to do the screening – all these numbers may, of course, change from week to week.

Repeating these calculations for successive weeks results in a ***working, quantified simulation*** of how the hiring process performs. The following diagram, for example, is a system dynamics model of the same hiring process above, but this model shows in the boxes (known as "stocks") how the ***number*** of applications at each stage change over 20 weeks. Those numbers change from week to week because applications "flow" from stage to stage, and it is the ***rates*** at which applications flow through the green "pipes" that determine how the numbers in each stock change from week to week.



You can explore this model and see how it performs under differing assumptions at sdl.re/HiringProcess.

What exactly is dynamic modeling doing? Quite different from process mapping, dynamic modeling focuses on ***the items*** of an enterprise themselves, so most of its elements are ***nouns*** – customers, dollars, units, people, products, and so on. (*In this simple example, the items are the job-applicants*). These models should also be rigorous. First, the stocks and flow-rates at the heart of a dynamic model are fundamental factors ***observable*** in the real world, and ***quantifiable***. (*See sdl.re/LIPstock for why these items are so important*). In addition, the behaviour of the model – all its elements, not just key results – should match observable behaviour of the real-world system.

Such dynamic models are ***entirely different*** in nature from process models and offer different benefits. They show how the system – whether just this local hiring policy or a whole enterprise – actually functions and performs, quantitatively and over time. And that performance encompasses ***everything*** in the system, not just end-results – the number of people hired in this case.