Towards the algorithmic detection of archetypal structures in system dynamics

Lukas Schoenenberger,[a]* Alexander Schmid[b] and Markus Schwaninger[c]

Abstract

Traditionally, model analysis follows qualitative, heuristic, and trial-and-error-driven approaches for testing dynamic hypotheses. Only recently have other methods like loop dominance analysis or control theory been proposed for this purpose. We advocate complementing established qualitative heuristics with a quantitative method for model analysis. To that end, we propose two algorithms to detect Wolstenholme's four generic problem archetypes within models. We tested these algorithms using the Maintenance and World Dynamics models. The approach presented in this paper is a first important step towards the identification of system archetypes in system dynamics and contributes to improving model analysis and diagnosis. Furthermore, our approach goes beyond diagnosis to eliciting solution archetypes, which foster the design and implementation of effective policies.

Syst. Dyn. Rev. **31**, 66–85 (2015)

Introduction

Over the last three decades, interest in developing formal tools for the detection of dominant structures in large system dynamics (SD) models has grown among system dynamicists (e.g. Kampmann and Oliva, 2009). While these tools have added substantially to our knowledge of how structure drives behavior in large models, they remain reserved for the more mathematically inclined scholars, and their use in the SD community is limited (ibid.). Therefore, we propose a new method for model analysis that is based on the *algorithmic detection of archetypal structures* (ADAS), an idea that can be traced back to Wolstenholme's (2003) seminal paper "Towards the definition and use of a core set of archetypal structures in system dynamics", winner of the 2004 Jay Wright Forrester Award (Andersen, 2004).

The ADAS method is a tool for testing a dynamic hypothesis of the following type: *An archetypal structure causes the dysfunctional behavior of a variable of interest*. More specifically, the ADAS method allows for the automated detection of all archetypal structures belonging to any of the four generic problem archetypes as defined by Wolstenholme (2003). Systematic

[a] Department of Business Administration, University of Zurich, Switzerland
[b] Institute of Information Systems, Vaduz, University of Liechtenstein, Liechtenstein
[c] Department of Management, University of St Gallen, St Gallen, Switzerland

* Correspondence to: Lukas Schoenenberger, Department of Business Administration, University of Zurich, Plattenstrasse 14, 8032 Zurich, Switzerland. E-mail: lukas.schoenenberger@business.uzh.ch; schoenenberger.lukas@bluewin.ch

Accepted by Andreas Größler; *Received 29 January 2014*; *Revised 23 April 2015*; *Accepted 27 April 2015*

System Dynamics Review
System Dynamics Review vol 31, No 1-2 (January-June 2015): 66–85
Published online 24 June 2015 in Wiley Online Library
(wileyonlinelibrary.com) DOI: 10.1002/sdr.1526

identification of all generic problem archetypes dramatically simplifies the formulation of effective policies because a solution archetype exists for each problem archetype (Wolstenholme, 2003). The method proposed in this paper is a contribution to model structure analysis, particularly to the rigor and effectiveness of SD-based model diagnosis for finding effective leverage points.

The ADAS method is probably most beneficial for the analysis of models that are not intentionally built around a specific problem archetype. This relates to the observation by Corben (1994, p. 16) that "in using the system archetypes for conceptualisation, there is a real danger that the selection of an archetype will be both a starting point and an ending point". Novice modelers in particular may select an inappropriate archetypal structure as a modeling basis or have a preconceived view of the problem because of using these structures. Thus the method presented here is useful for modelers who begin the modeling process from scratch and engage in the task of structuring their own view of a system without reverting to the use of archetypal structures as model templates (ibid.).

This paper is organized into eight interrelated sections. Following the introduction, the second section presents a methodological overview and embeds the ADAS method in a generic process of model building and analysis. The third section discusses the application of the ADAS method to a small model in order to provide the reader with a better sense of the proposed method's value. The fourth section introduces the background necessary to the ADAS method. The fifth section presents the algorithms to detect the four generic problem archetypes. The sixth section applies the ADAS method to a substantially more complex model—Forrester's (1971) classical World Dynamics model—and suggests an effective heuristic for reducing the large number of archetypal structures detected in this model. The seventh section covers the limitations of the ADAS method, and the last section provides conclusions and recommendations for future research.

Methodological overview

The ADAS method builds on the assumption that the structure of an SD model can be accurately described as a directed graph (Oliva, 2004; Kampmann, 2012). This implies that variables and relationships in SD models are transcribed into vertices and edges, respectively. The graph representation of system structure is an indispensable requirement to algorithmically check SD models for the presence of archetypal structures. However, for the detection of generic problem archetypes, more information is necessary beyond the mere notion of connectivity of variables. More specifically, the algorithms need two additional parameters as inputs: the polarity of relationships and the existence and magnitude of delays. To provide the algorithms with this additional information, we propose a qualitative coding procedure.

DOI: 10.1002/sdr

A first set of edge weights $e_{i,j}=\{1, -1\}$ is used to distinguish between "positive" (same direction) and "negative" (opposite direction) causal relationships between independent and dependent variables. A second set of edge weights $\tau_{i,j}=\{1, 2, 4\}$ serves to discriminate between delayed and non-delayed causal effects. For both categories of edge weights $e_{i,j}$ and $\tau_{i,j}$, information is stored in an adjacency matrix. A more detailed description of this coding procedure can be found in the section entitled "Technical background to the ADAS method". Due to the structural equivalence of three generic problem archetypes, only two algorithms are needed to detect all four archetypal structures: underachievement, relative achievement, relative control and out-of-control archetypes. Each archetype is composed of an intended consequence (ic) feedback loop and an unintended consequence (uc) feedback loop. A thorough discussion about the algorithms is presented in the section on "Algorithms to detect Wolstenholme's four generic problem archetypes". In the following, we demonstrate how the ADAS method can be embedded in a generic process of model building and analysis.

Table 1 clarifies the basic steps of the entire model analysis process with the ADAS method integrated. Steps (1) and (2) are in line with traditional practices in SD: identify an undesirable system behavior and formulate a "theory" (dynamic hypothesis) about how the system creates the troubling behavior (Forrester, 1994). Steps (3)–(8) are specific to this model analysis process, while step (5)—the application of the ADAS method—plays an especially important role.

Application of the ADAS method to a small model

This section describes the generic model building and analysis process step by step to demonstrate and explain how the algorithms might be applied to

Table 1. Generic model building and analysis process that integrates the ADAS method

Step	Description
1	Identify a problematic reference behavior of a variable of interest
2	Formulate a dynamic hypothesis (model), in terms of a stock and flow diagram (SFD) for why this dysfunctional behavior occurs
3	Convert the SFD into a directed graph by using two adjacency matrices, the first indicating link polarities and the second covering delays
4	Set the variable of interest identified in step (1) as the outcome variable (key variable) for the algorithms
5	Check algorithmically if the variable of interest is part of one or several archetypal structures (ADAS method)
6	Identify plausible archetypal structures that cause the problematic behavior of the variable of interest. To that end, reinterpret the found archetypes in step (5) in the context of the model built in step (2)
7	Introduce solution links (policies) as suggested in the literature (Wolstenholme, 2003)
8	Simulate the model and review the behavior of the variable of interest

DOI: 10.1002/sdr

a small model. The application shows how they can significantly improve system diagnosis as well as the discovery and implementation of policies.

Figure 1 shows the Maintenance Model (Thun, 2006; Sterman, 2000), which illustrates the problematic and typical behavior of a production system where reactive maintenance predominates, maintenance that is breakdown induced, triggering an undesired growth of equipment defects. In this example, rising equipment defects—the reference behavior—are the starting point for the model-building and analysis process. As suggested by Wolstenholme (2004), we added two redundant relationships connecting the two outflows back to the stock (bold lines) to better visualize the balancing feedback loops in Figure 1. These modifications are for illustrative purposes only and facilitate the drawing of the digraph.

Next, information about link polarities and time delays are stored in two adjacency matrices. In the polarity matrix, "1" stands for two variables being positively related and "−1" stands for a pair of variables being negatively related. All the information needed for this task can be directly deduced from Figure 1. In the temporality matrix, we use "1" to indicate no significant time delay between variable pairs, "2" for all links emerging from the stock variable and "4" if prominent delays exist. We assume that all links emerging from a stock variable are slightly delayed following the argument of stocks being the sources of delays in SD models (Sterman, 2000). According to Thun (2006), only one relationship has a significant delay and is therefore coded with a "4"; it is the link between *Takedown Rate* (variable 10) and *Preventive Maintenance* (variable 12). Table 2 presents these two adjacency matrices, accounting for

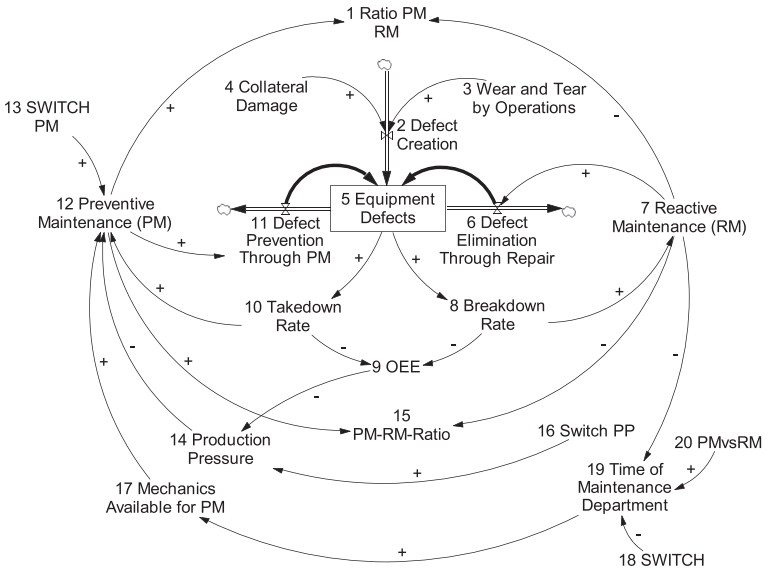


Fig. 1. The basic Maintenance Model (Thun, 2006, p. 167). The bold lines are included for illustrative purposes only

DOI: 10.1002/sdr

Table 2. Adjacency matrices for link polarity (left) and temporality (right)

Polarity matrix																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2				1																
3	1																			
4	1																			
5						1	1													
6				−1																
7	−1			1											−1				−1	
8					1		−1													
9														−1						
10							−1			1										
11				−1																
12	1								1					1						
13											1									
14											−1									
15																				
16													1							
17											1									
18																	−1			
19																1				
20																	1			

Temporality matrix																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2					1															
3		1																		
4		1																		
5								2	2											
6					1															
7	1			1											1				1	
8						1	1													
9														1						
10								1			4									
11					1															
12	1									1					1					
13											1									
14											1									
15																				
16														1						
17											1									
18																			1	
19																	1			
20																			1	

the polarity and temporality of each link. The numbering in the two matrices corresponds to the variable numbers in Figure 1.

In this example, variable 5, *Equipment Defects*, is the outcome variable and is algorithmically checked for membership in archetypal structures. Additionally, variable 7, *Reactive Maintenance*, is the control action for the detection of out-of-control archetypes.[1] The reasoning behind the choice of *Reactive Maintenance* as an input (control action) for the out-of-control algorithm is twofold: first, the amount of *Reactive Maintenance* is directly controllable by a manufacturing company; and second, the existence of an out-of-control archetype in this particular case is documented in the literature (Thun, 2006). Table 3 shows the results of this analysis and exhaustively lists all archetypal structures found (according to the definition of Wolstenholme, 2003), without any manual or other form of pre-selection.

In this model, the current control mechanism is not effective at stabilizing or reducing *Equipment Defects*. Consequently, the algorithm finds an out-of-control archetype wherein the intention to control *Equipment Defects* with *Reactive Maintenance* (variable 7) results in an unintended systemic reaction that exacerbates the problem in the long run. More *Reactive Maintenance*

[1]The out-of-control detection algorithm requires the definition of a *control action* as an input (see section about "Algorithms to detect Wolstenholme's four generic problem archetypes").

DOI: 10.1002/sdr

Table 3. Algorithmically detected generic archetypal structures

Intended consequence (ic)				Unintended consequence (uc)		
Underachievement archetype						
No.	D	P	Loop	D	P	Loop
1	7	R	5 → 10 → 9 → 14 → 12 → 11 → 5	5	B	5 → 8 → 7 → 6 → 5
Out-of-control archetype						
No.	D	P	Loop	D	P	Loop
1	5	B	5 → 8 → 7 → 6 → 5	8	R	5 → 8 → 7 → 19 → 17 → 12 → 11 → 5
Relative control archetype						
No.	D	P	Loop	D	P	Loop
1	5	B	5 → 8 → 7 → 6 → 5	8	B	5 → 10 → 12 → 11 → 5
2	8	B	5 → 10 → 12 → 11 → 5	5	B	5 → 8 → 7 → 6 → 5

D, delay; P, loop polarity; B, balancing; R, reinforcing.

reduces the *Time of the Maintenance Department* (variable 19) for other activities, resulting in fewer *Mechanics Available for Preventive Maintenance* (variable 17). In turn, lower execution of *Preventive Maintenance* (variable 12) reduces *Defect Prevention through Preventive Maintenance* (variable 11) and boosts *Equipment Defects* (variable 5).

Wolstenholme (2003, p. 12) suggests that "the closed loop solution to an out-of-control archetype lies in introducing or emphasising a direct link (the 'solution link') between the problem and the system reaction. The purpose of this link is to introduce or re-emphasise a further balancing loop in parallel with the ic balancing loop to counter the reinforcing reaction." Consequently, to address the archetypal behavior in the Maintenance Model, a direct relationship between the problem (*Equipment Defects*) and the

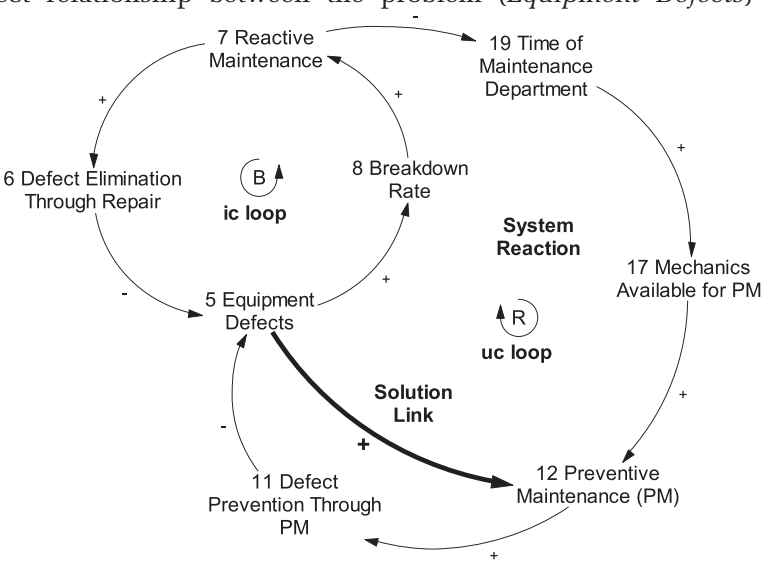


Fig. 2. Out-of-control archetype and suggested solution link

DOI: 10.1002/sdr

system reaction (*Preventive Maintenance*) is necessary. Figure 2 depicts the out-of-control archetype found in the model and the suggested solution link (bold arrow).

The implementation of this new solution link has a strong positive effect on equipment defects that is significantly reduced compared to the initial model (Figure 3). The simulation results in Figure 3 are generated using Thun's (2006) original model specifications. Figure 3 demonstrates the difference between the out-of-control archetype in the original model and the model with our suggested solution link: equipment defects grow rapidly and stabilize at a very high point in the original model, while they are much smaller and stabilize much more quickly once the solution is implemented.

Technical background to the ADAS method

This section provides the background to the ADAS method to help the reader understand more complex applications, and also provides proper technical documentation of the method. It will be shown how SD models can be described as directed graphs (digraphs) and how relational information, such as link polarity and delay, can be stored in adjacency matrices. The conversion of a simulation model to a digraph is a necessary precondition for the efficient application of our algorithms, presented in the next section.

In mathematical notation, an SD model can be described as a digraph G consisting of a set of vertices V and a set of edges E. Every edge connecting two vertices v_i, $v_j \in V$ (i.e. $v_i \rightarrow v_j = e_{i,j} \in E$) denotes a direct causal relationship. In accordance with Oliva (2004), Kampmann (2012)

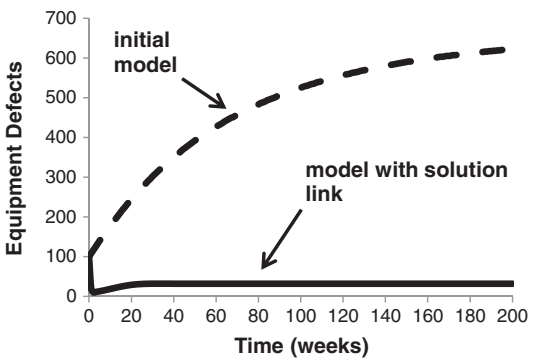


Fig. 3. Comparison between *Equipment Defects* in the initial model by Thun (2006) and in the revised model with Wolstenholme's (2003) solution link

DOI: 10.1002/sdr

and Schaffernicht and Groesser (2014), we use the following edge weights $e_{i,j}$ to account for link polarity:

$$e_{i,j} = \begin{cases} -1 \text{ if } v_i \xrightarrow{-} v_j \\ +1 \text{ if } v_i \xrightarrow{+} v_j \end{cases}$$

In graph notation, $v_i \xrightarrow{-} v_j$ indicates that a change in v_i causes v_j to change in the opposite direction (i.e. if v_i increases then v_j decreases, and vice versa), which corresponds to a negative link polarity in SD models. In contrast, $v_i \xrightarrow{+} v_j$ implies that a change in v_i causes v_j to change in the same direction (i.e. if v_i increases/decreases then v_j also increases/decreases), which corresponds to a positive link polarity in SD models.

Links in SD models are also characterized by the presence and magnitude of delays (Sterman, 2000). To incorporate temporal information in a digraph, we propose an additional set of edge weights $\tau_{i,j}$ building on Schaffernicht and Groesser (2014):

$$\tau_{i,j} = \begin{cases} 1 \text{ if a change in } v_i \text{ immediately affects } v_j \\ 2 \text{ if } v_i \text{ is a stock variable} \\ 4 \text{ if a change in } v_i \text{ affects } v_j \text{ only after a significant delay} \end{cases}$$

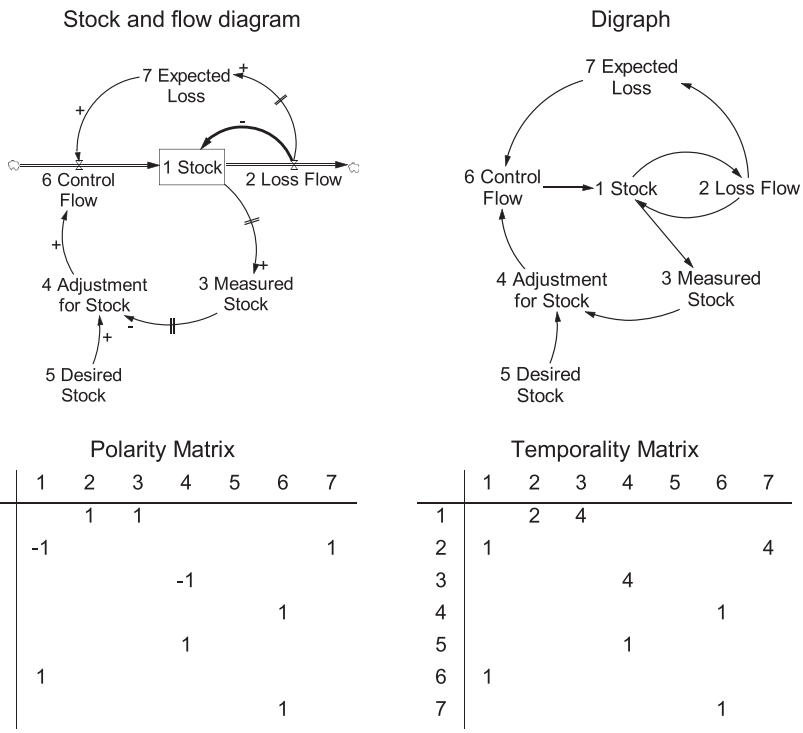
The time delay between a change in vertex v_i and its effect on v_j is indicated by $\tau_{i,j}$. If a change in v_i immediately impacts v_j then $\tau_{i,j} = 1$. If v_i is a stock variable then we assume that impacts on all v_j are slightly delayed, defining $\tau_{i,j} = 2$. This assumption follows the argument of stocks being the sources of delays in SD models (Sterman, 2000). Additionally, if a change in v_i impacts v_j only after a significant delay, then $\tau_{i,j} = 4$.

The relational information—link polarities and delays—can be stored in adjacency matrices. An adjacency matrix A is a $|V| \times |V|$ square matrix with $A = (e_{i,j})$ (Cormen *et al.*, 2009). Each vertex appears twice, both in a row and a column. The values in row A_i represent the successor set for vertex i, while the values in column A_i represent the predecessor set for vertex i. Figure 4 shows the example of a stock and flow diagram (SFD), an SFD as a digraph and the corresponding adjacency matrices. Oliva (2004) uses a very similar procedure for representing SD models as digraphs.

In graphs, a directed path is a series of disjoint (distinct) connected vertices. Like edges, paths are made up of characteristic attributes—in this case polarity, temporality, and length. The polarity of a path is calculated by multiplying the path's edge polarities (Kampmann, 2012; Richardson, 1995); temporality is calculated by the addition of time delays;[2] and length by the number of edges in the path (Freeman *et al.*, 1991).

[2]The adequacy of adding time delays can be demonstrated by means of a simple example of two edges: $v_i \rightarrow v_{i+1}$ and $v_{i+1} \rightarrow v_i$. Each vertex impacts its successor after a temporal unit x. Thus a change in vertex v_i influences v_{i+1} after a time delay x and the same holds for $v_{i+1} \rightarrow v_i$. Consequently, it takes in total $x + x = 2x$ units of time for a change in v_i to feed back into the loop $v_i \rightarrow v_{i+1} \rightarrow v_i$.

DOI: 10.1002/sdr



Polarity Matrix

	1	2	3	4	5	6	7
1		1	1				
2	-1						1
3				-1			
4						1	
5				1			
6	1						
7						1	

Temporality Matrix

	1	2	3	4	5	6	7
1		2	4				
2	1						4
3				4			
4						1	
5				1			
6	1						
7						1	

Fig. 4. SFD with corresponding digraph and adjacency matrices. For illustrative reasons, we assume that three links exhibit a significant delay and we mark these three links with a double bar in the SFD. Furthermore, we add one redundant link connecting the loss flow back to the stock (bold line) to better visualize one balancing feedback loop in the SFD

Paths forming loops indicate feedback effects in models. Loops are paths with distinct edges and only one vertex appearing twice: the first and the last vertex (Tarjan, 1972). In graph notation, a path forms a loop when $v_n = v_i$ is true for $v_i \rightarrow v_{i+1} \rightarrow \ldots \rightarrow v_n$. Feedback loops are the core elements of archetypal structures.

Algorithms to detect Wolstenholme's four generic problem archetypes

In this section, we present two algorithms for the detection of the four generic problem archetypes described by Wolstenholme (2003): the underachievement, relative achievement, relative control and out-of-control archetypes. Owing to the structural equivalence of three generic problem archetypes, only two algorithms are needed to identify all four archetypal structures.

Detecting underachievement, relative achievement and relative control archetypes

The structures of the underachievement, relative achievement and relative control archetypes vary only with respect to the polarities of the intended

DOI: 10.1002/sdr

(ic) and unintended consequence (uc) loops. Figure 5 illustrates the basic structure of these three archetypes in a causal loop diagram (CLD). Vertex v_o represents the outcome variable that is intentionally driven by an action variable v_a. The system reaction v_r that unfolds only after a certain delay, however, compromises the outcome variable v_o.

To detect whether this basic structure is present in a model, every ic- and uc-loop combination that has no intersection other than the outcome variable v_o must be identified. Therefore, the algorithm described in Figure 6 processes a polarity adjacency matrix A and an outcome variable of interest v_o as input parameters. Furthermore, using two Boolean parameters ($bool_{ic}$, $bool_{uc}$), the polarity of the ic- and uc-loops to be identified can be set.

The Boolean parameters ($bool_{ic}$, $bool_{uc}$) are set to be true when the loops are reinforcing and false otherwise. The call of the algorithm then decides which archetypal structure is returned:

For the underachievement archetype:
FINDARCHETYPES(A, v_o, true, false)

For the relative achievement archetype:
FINDARCHETYPES(A, v_o, true, true)

For the relative control archetype:
FINDARCHETYPES(A, v_o, false, false)

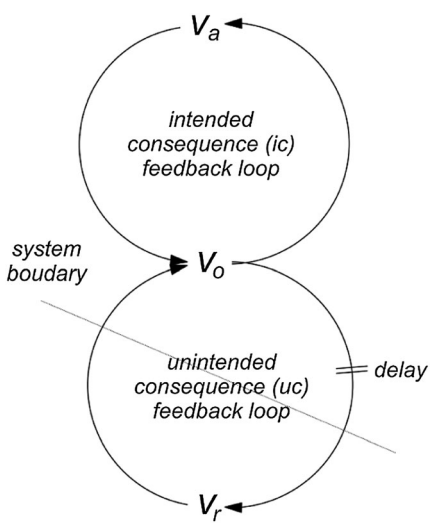


Fig. 5. Basic common structure of underachievement, relative achievement and relative control archetypes

DOI: 10.1002/sdr

```
1   # INPUT: Adjacency matrix A, vertex v_o ∈ V(A), two Boolean expressions each {true ∨ false}
2   # OUTPUT: a set of archetypal structures
3
4   function FINDARCHETYPES(A, v_o, bool_ic,, bool_uc)
5       T                                          ◁ set of archetypes in the form (ic loop, uc loop)
6       Λ_ic , Λ_uc ← ∅                            ◁ sets of loops
7       for each path p ∈ FINDPATHS(v_o, v_o, ∅)   ◁ for all loops containing v_o
8           if (Polarity[p] > 0) == bool_ic        ◁ if polarity matches parameter
9               Λ_ic ← Λ_ic ∪ p                    ◁ add path to set of loops
10          end if
11          if (Polarity[p] > 0) == bool_uc        ◁ if polarity matches parameter
12              Λ_uc ← Λ_uc ∪ p                    ◁ add path to set of loops
13          end if
14      end for
15      for each l_ic ∈ Λ_ic                       ◁ for all potential ic loops
16          for each l_uc ∈ Λ_uc                   ◁ for all potential uc loops
17              i ← false                          ◁ intialize intersection check
18              for all n ∈ l_ic                   ◁ for all nodes in potential ic loop
19                  if (n ≠ v_o) ∧ (n ∈ l_uc)      ◁ if any other vertex than v_o is contined in uc loop
20                      i ← true                   ◁ intersection is true
21                      break                      ◁ break for loop
22                  end if
23              end for
24              if not i                           ◁ if ic- and uc-loop do not intersect
25                  T ← T ∪ (l_ic, l_uc)           ◁ add loop combination to archetype set
26              end if
27          end for
28      end for
29      FINDARCHETYPES ← T                         ◁ asign set of archetypes as function value
30  end function
```

Fig. 6. Algorithm to detect underachievement, relative achievement and relative control archetypes

First, the algorithm stores all loops containing the outcome variable v_o and meeting the polarity criteria (e.g. reinforcing/reinforcing for the relative achievement archetype) in two separate lists. Second, each ic-loop (l_i) is compared with every uc-loop (l_u). If they do not intersect, meaning they have no other variable in common except v_o, the l_i / l_u-loop combination is added to the final archetype list returned by the algorithm.

The resulting list with archetypal structures might be long and difficult to interpret in large models (Kampmann, 2012). Therefore, the choice of v_o is of crucial importance. We suggest focusing on vertices or loops with high relevance to the model. Vertex and loop relevance could be approximated, for example, by centrality measures (Oliva, 2004; Schoenenberger and Schenker-Wicki, 2014; Wunderlich *et al.*, 2014) and dominance concepts respectively (Borgatti and Everett, 2006; Freeman *et al.*, 1991; Richardson, 1995).

Detecting out-of-control archetypes

The fourth two-loop system archetype by Wolstenholme (2003)—the out-of control-archetype—comprises a balancing ic-loop and a reinforcing uc-loop. The ic-loop is meant to control the magnitude of a problem. However, the uc-loop creates a reinforcing loop, resulting in a worsening of the problem that

DOI: 10.1002/sdr

might run out-of-control. Rather than the outcome itself, it is the controlling action that provokes the detrimental system reaction.

Accordingly, the two-loop structure is slightly different from the other three generic archetypes. Figure 7 depicts the CLD for the out-of-control archetype.

In this archetypal structure, the ic- and uc-loops partly overlap, so a different algorithm is needed from that in Figure 6. An appropriate algorithm for identifying out-of-control archetypes is shown in Figure 8. Again, the algorithm requires inputs of an adjacency matrix A and an outcome variable v_o. Additionally, a parameter for the control action v_c is required to limit the resulting archetype set and to facilitate the interpretation of the output.

First, the algorithm stores all balancing ic-loops that contain the outcome variable v_o and the control variable v_c in a data collection. Second, all paths from the control variable v_c to the outcome variable v_o are stored in another dataset as potential *system reaction paths*. Third, the algorithm iterates through all potential ic-loops and divides them into two paths, both of which exclude the control variable v_c (see Figure 7). They are: (1) a path preceding the control variable v_c; and (2) a path succeeding the control variable v_c. The algorithm now iterates through all potential system reaction paths. It checks to ensure that the potential system reaction path does not intersect with the two previously defined paths, and that it includes a potential system reaction variable (length > 2). If these criteria are fulfilled, the polarities are checked and the loop combination is added to the resulting archetype set.

To fit the out-of-control archetype from Wolstenholme's (2003, p. 17) typology, the polarity of the path from the outcome variable v_o to the control action v_c has to match the polarity of the path from the control action v_c via the system reaction path to the outcome variable v_o. If this criterion is fulfilled, the structure potentially matches the out-of-control archetype and will be

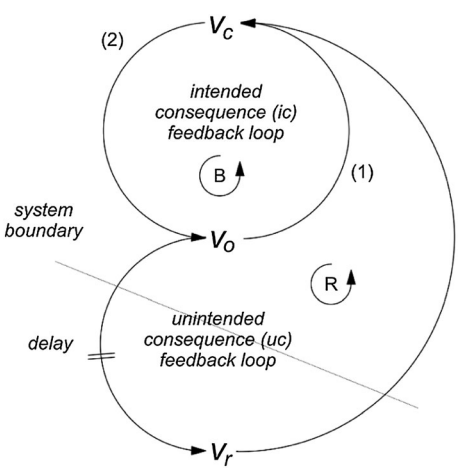


Fig. 7. Out-of-control archetype

DOI: 10.1002/sdr

Fig. 8. Algorithm to detect out-of-control archetypes

```
1  INPUT: Adjacency matrix A and vertices {v_o , v_c} ∈ V(A)
2  # OUTPUT: a set of archetypal structures of the type Out-of-Control
3
4  function FINDOOCARCHETYPES(A, v_o, v_c)
5      T                                          ◅ set of archetypes in the form: (ic loop, uc loop)
6      Λ_ic , P_uc , P_r ← ∅                      ◅ initialize sets of paths
7      for each path p ∈ FINDPATHS(v_o, v_o, ∅)   ◅ for all loops containing v_o
8          if (Polarity[p] < 0) ∧ (v_c ∈ p)       ◅ if polarity is negative and v_c is part of loop
9              Λ_ic ← Λ_ic ∪ p                    ◅ add path to set of potential ic loops
10         end if
11     end for
12     P_r ← FINDPATHS(v_c, v_o, ∅)               ◅ store all paths from v_c to v_o
13     for each l_ic ∈ Λ_ic                       ◅ for all loops in set of potential ic loops
14         p_bc , p_ac ← ∅                        ◅ initialize two new paths (path before and after v_c)
15         a ← false                              ◅ initialize boolean check whether variable in path is after v_c
16         for each n ∈ l_ic                      ◅ for all vertices in potential ic loop
17             if n ≠ v_o                         ◅ if vertex is not v_o
18                 if not a                       ◅ if check is not true (vertex is not after v_c)
19                     if n ≠ v_c                 ◅ if vertex is not v_c
20                         Enqueue(p_bc, n)       ◅ add vertex to p_bc
21                     else
22                         a ← true               ◅ set check true indicating next vertex to be after v_c
23                     end if
24                 else
25                     Enqueue(p_ac, n)           ◅ add vertex to p_ac (path after control variable)
26                 end if
27             end if
28         end for
29         for each r ∈ P_r                       ◅ for all paths from v_c to v_o
30             i ← false                          ◅ initialize intersection check
31             for each n ∈ p_bc                  ◅ for all vertices in p_bc
32                 if n ∈ r                       ◅ if vertex n is part of r
33                     i ← true                   ◅ p_bc and r intersect
34                     break                      ◅ break
35                 end if
36             end for                            ◅ next vertex in p_bc
37             if not i                           ◅ if p_bc and r do not intersect
38                 for each n ∈ p_ac              ◅ for all vertices in p_ac
39                     if n ∈ r                   ◅ if vertex n is part of r
40                         i ← true               ◅ p_ac and r intersect
41                         break                  ◅ break
42                     end if
43                 end for                        ◅ next vertex in p_ac
44             end if
45             if (not i) ∧ (Length[r] > 2)       ◅ if r does not intersect with p_bc and p_ac
                                                      and contains more than only v_c and v_o
46                 c ← ∅                          ◅ initialize an empty path
47                 Enqueue(c, v_o)                ◅ add vertex v_o to c
48                 Enqueue(c, p_bc)               ◅ add vertex p_bc to c
49                 Enqueue(c, v_c)                ◅ add vertex v_c to c
50                 if Polarity[c] == Polarity[r]  ◅ if polarity of paths is the same
51                     l_uc ← ∅                   ◅ initialize uc loop l_uc
52                     Enqueue(l_uc, v_o)         ◅ add vertex v_o to l_uc
53                     Enqueue(l_uc, p_bc)        ◅ add vertices from path before control (p_bc) to l_uc
54                     Enqueue(l_uc, r)           ◅ add r to l_uc so that
55                     T ← T ∪ (l_ic, l_uc)       ◅ add loop combination to archetype set
56                 end if
57             end if
58         end for
59     end for
60     FINDOOCARCHETYPES ← T                      ◅ assign set of archetypes as function value
61 end function
```


DOI: 10.1002/sdr

returned by the algorithm in a collection with other potential archetypal structures.

Application of the ADAS method to a large model

While the application of this method to a small model was introduced earlier, the following large application is only described after introducing the complete technique of the ADAS method in order to facilitate the reader's understanding of this extended case. We will now apply the ADAS method to Forrester's (1971) World Dynamics model. This way, the functioning of the algorithms should become clearer, and the interpretations of the computational outcomes become more amenable to insights. In particular, we discuss an effective heuristic for the reduction of the large number of archetypal structures identified in the World Dynamics model.

To test the two proposed algorithms, we have adapted Forrester's (1971) World Dynamics model by eliminating the lookup variables (time tabs).[3] The removal of these exogenous variables has no effect on the outcome of the ADAS analysis, as they are not part of any feedback loop. The adapted model contains 59 variables and 88 links. In this model, population growth —which peaks in the year 2020 before declining—is the reference mode.

We compiled both adjacency matrices following the coding procedure explained earlier in this paper. The information for the polarity matrix can be directly extracted from the model. For the compilation of the temporality matrix, we control for significant delays. We assume three relationships in the model to be significantly delayed: (1) the impact of *Pollution Absorption Time* (variable 58) on *Pollution Absorption* (variable 59); (2) the impact of *Capital Investment from Quality Ratio* (variable 35) on *Capital Agriculture Fraction* (variable 36); and (3) the impact of *Capital Agriculture Fraction Indicated* (variable 22) on *Capital Agriculture Fraction* (variable 36). The first relationship exhibits a significant delay because pollution is not absorbed immediately but only after a substantial time delay. Forrester (1971) explicitly integrated this delay through the variable *Pollution Absorption Time*. The latter two relationships are significantly delayed as the *Capital Agriculture Fraction* (variable 36) needs time to adjust to changes, which is represented by the *Capital Agriculture Fraction Adjustment Time* (variable 37) in the model.

Population (variable 1) is set as the variable of interest and algorithmically checked for being part of archetypal structures. Owing to the high number of archetypal structures detected, we only focus on underachievement archetypes; these are the most relevant structures for describing the reference behavior of population growth and decline. The underachievement archetypes in this model consist of a reinforcing ic-loop that drives population growth and a balancing uc-loop

[3]A list of the variables used for the algorithms can be found in the electronic supplement, provided as supporting information. Additionally, both matrices, polarity and temporality, and all detected underachievement archetypes can be retrieved from the electronic supplement.

DOI: 10.1002/sdr

that slows growth, usually as a result of a resource constraint (Wolstenholme, 2003). These structures are promising candidates for explaining the deceleration of population growth in the mid 20th century (in the model), but they cannot tprovide insights into the causes for population decline starting in the year 2020.

Running the detection algorithm over the model, a total of 99 underachievement archetypal structures are returned. Obviously, the high number of potentially archetypal structures hinders modelers from efficiently finding and recommending favorable policies. Therefore, we recommend working with a reduced loop set instead of algorithmically processing *all* feedback loops provided by the model. In the SD literature, three different types of reduced loop sets are proposed: (1) independent loop set (ILS) (Kampmann, 2012); (2) shortest independent loop set (SILS) (Oliva, 2004); and (3) minimal shortest independent loop set (MSILS) (Oliva, 2004). All of them considerably reduce the feedback complexity of SD models. In this article, we focus on the SILS and MSILS. This follows the advice of Oliva (2004, p. 331): "The fact that feedback complexity can be reduced to a unique granular representation of independent feedback loops leads me to posit that the SILS and MSILS should be considered a basis for our field's efforts to understand loop dominance."

Applying Oliva's (2004) SILS algorithm to the World Dynamics model results in a 70 percent reduction in the total number of feedback loops; the 80 loops in the initial model are reduced to 24 loops in the SILS. Using only the SILS as an input for the proposed algorithm diminishes the resulting archetype set to 20 underachievement archetypal structures (see Table A.1 in the Appendix). In contrast, adopting Oliva's (2004) MSILS algorithm to the same model generates a 75 percent reduction in the total feedback loop number; the 80 loops in the original World Dynamics model are reduced to 20 loops in the MSILS. However, using MSILS as an input for our algorithm only leads to a minimal difference in terms of detected underachievement archetypal structures compared to SILS. More specifically, the MSILS produces the same archetypal solution set except for one archetypal structure (see archetypes marked with one asterisk in Table 1.A in the Appendix).

To show the interaction between the algorithm's operation and subsequent human interpretation, we will briefly delve into the 20 archetypal structures identified from the SILS. For this end, we define policy variables that can achieve the intended outcome of population growth and check whether they entail any unintended consequences. For example, one possible means of boosting population is to enact policies that raise the birth rate (*Births*, variable 15). If this is done, then the first seven structures show seven different unintended consequences that arise when increasing *Births* is used to boost *Population*. All of these unintended consequences eventually lead to more *Deaths* (variable 2), thereby counteracting the intended consequence loop. Another policy approach to generating population growth might be increasing *Capital Investment* (variable 25). *Capital Investment* leads eventually to more money spent on agriculture (*Capital Ratio Agriculture*, variable 32), which in

DOI: 10.1002/sdr

turn increases both the *Food Ratio* (variable 17) and the birth rate (*Births*, variable 15). In that case, archetypal structures 17–20 are worth a look because they show four different unintended consequences of increasing *Capital Investment*. Figure 9 illustrates these four archetypal structures that arise when *Capital Investment* is the policy approach.

Wolstenholme (2003, p.12) proposes "that the closed-loop solution to an underachievement archetype lies in trying to use some element of the achievement action to minimise the reaction in other parts of the organisation, usually by unblocking the resource constraint. That is to introduce a further reinforcing loop in parallel with the ic reinforcing loop to counter the balancing reaction." If we transfer Wolstenholme's idea to the policy example of increased *Capital Investment* (variable 25), this means that capital investments should be used to minimize the unintended consequences. For example, this suggests investing capital in the reduction of the *Death Rate* (variable 2) for archetypal structures 17 and 18, investing capital in the reduction of the *Pollution Generation* (variable 46) for archetypal structure 19, and investing capital in the reduction of the *Natural Resource Utilization* (variable 54) for archetypal structure 20. More specifically, we ought to introduce three solution links with negative polarity from *Capital Investment* (variable 25) to *Death Rate* (variable 2), from *Capital Investment* (variable 25) to *Pollution Generation* (variable 46), and from *Capital Investment* (variable 25) to *Natural Resource Utilization* (variable 54).

The implementation of these three solution links shows logically plausible results: each policy has a strong impact on population growth compared to the

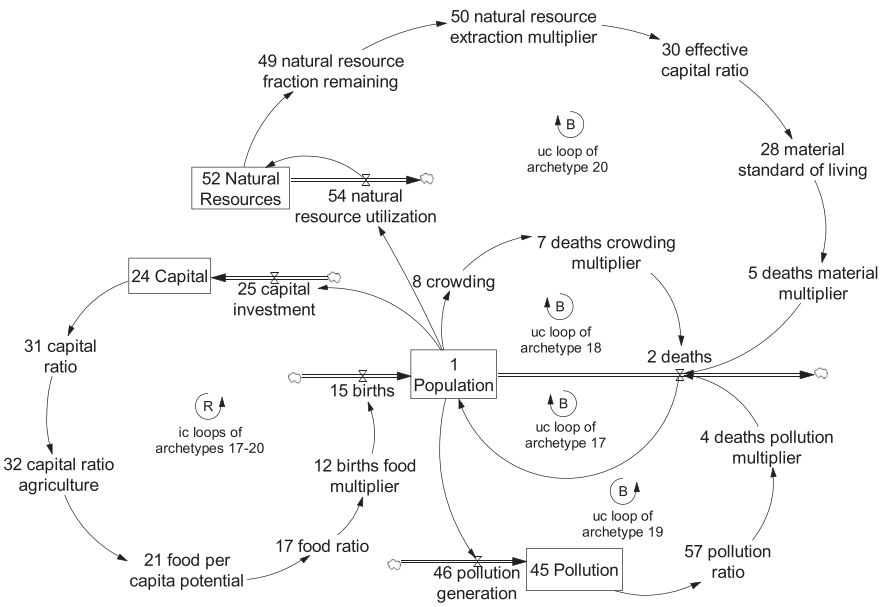


Fig. 9. Archetypal structures resulting from a policy of more *Capital Investment*

DOI: 10.1002/sdr

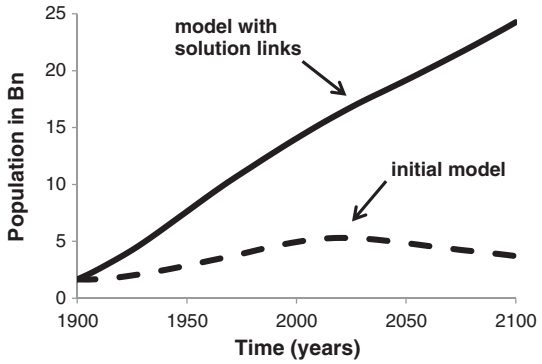


Fig. 10. Comparison between population growth in the initial model by Forrester (1971) and in the revised model with Wolstenholme's (2003) solution links

initial model and leads to an almost linear growth until the year 2100 (Figure 10). The simulation results in Figure 10 are achieved using Forrester's (1971) original model specifications.

Limitations

The ADAS method is new and has limitations that prescribe future research avenues. First, this method cannot yet detect system archetypes, only modeling components that fulfill the structural requirements to qualify as system archetypes. True archetypes are more than simple two-loop constellations; they are real-world phenomena with causes and effects separated in time and space (Wolstenholme, 2003). In particular, spatial differences between ic- and uc-loops delineated by system boundaries are not incorporated in the ADAS method. Therefore, modelers' judgments are required for the interpretation of this analysis, preventing fully automated archetype detection.

Second, the output of the algorithms might be difficult to interpret in large models because of the current lack of criteria other than structural requirements for the identification of system archetypes. The analysis of the World Dynamics model illustrated this problem: the ADAS method found many structures that are potentially archetypal. This problem recurs in any large model because the probability for detecting archetypal structures rises as the number of feedbacks grows. In reality however, not all of the detected structures are plausible explanations for counterintuitive system behaviors.

Discussion and conclusions

The ADAS method is a major step towards the automated identification of system archetypes in SD models. It is highly specific and straightforward in its application because it focuses model analysis on the specific question: Is a certain variable of interest with dysfunctional reference behavior part of an

DOI: 10.1002/sdr

archetypal structure in the model? Answering this question can substantially improve systematic diagnosis as well as the discovery and implementation of structural changes (via solution links) that mitigate or even reverse the problematic behavior. We see the strength of this approach in its complementarity to established SD practices—in particular to the eigenvalue elasticity analysis (EEA)—and its potential to significantly shorten the process of model analysis.

The ADAS method in its current form has some restrictions, as addressed earlier. In particular, analyzing models of high feedback complexity causes the ADAS method to return a large amount of potential archetypal structures. As suggested previously, this problem can be effectively tackled by working from a reduced loop set such as the SILS. Additionally, using the SILS in combination with the ADAS method makes the results comparable and complementary to the EEA, which by necessity focuses on these loop sets. Future research might try to integrate the SILS algorithm into the ADAS method, which would allow for a more effective analysis of "big" SD models.

Furthermore, the introduction of an additional coding procedure—besides the qualitative code for link polarities and time delays—to account for spatial effects in system archetypes might also be a promising endeavor for future research. This could be a set of edge weights $\theta_{i,j} = \{0,1\}$ that distinguishes between variable pairs being within the same organizational "compartment" and variable pairs being separated by organizational boundaries. Based on such a code, the algorithm would be able to recognize organizational boundaries in models and integrate them into the process of system archetype detection. In conclusion, despite current limitations and the nascent status of this method, the proposed algorithms represent an important step forward for model analysis and a pathway to the identification of "real" system archetypes in SD.

Acknowledgements

The authors appreciate the technical support of Professor Rogelio Oliva with respect to the calculation of the SILS and MSILS. We are equally thankful to Duke-NUS/SingHealth Academic Medicine Research Institute and the medical editing assistance of Taara Madhavan (Associate, Clinical Science, Duke-NUS Graduate Medical School). Furthermore, we are very grateful for the valuable feedback of Professor Andreas Grössler and of two anonymous reviewers.

Biographies

Lukas Schoenenberger is a research associate at the University of Zurich, Switzerland, and a postdoctoral research fellow at the Duke–NUS Graduate Medical School, Singapore. His academic work revolves around managing complexity in management and healthcare through qualitative and quantitative modeling. In his research he combines a variety of approaches such as system

DOI: 10.1002/sdr

dynamics, group model building, social network analysis, graph theory and stakeholder analysis.

Alexander Schmid is a PhD student at the University of Liechtenstein, Principality of Liechtenstein, and works as a research analyst for the software company SAP. He studied management, informatics and biology at the University of Zurich and at the University of Bern, Switzerland.

Markus Schwaninger is a Professor Emeritus of Management at the University of St. Gallen, Switzerland. His research focus is on complex dynamic systems, namely organizational design, transformation and learning, as well as the methods of model-based management. His methodological emphasis is on system dynamics and organizational cybernetics.

References

Andersen D. 2004. The 2004 Jay Wright Forrester Award: citation for the winner. Eric F. Wolstenholme. *System Dynamics Review* **20**(4): 337–440.

Borgatti SP, Everett MG. 2006. A graph-theoretic perspective on centrality. *Social Networks* **28**(4): 466–484.

Corben DA. 1994. Integrating archetypes and generic models into a framework for model conceptualisation. In Proceedings of the 1994 International System Dynamics Conference. System Dynamics Society: Stirling, UK.

Cormen TH, Leiserson CE, Rivest RL, Stein C. 2009. *Introduction to Algorithms*, (3rd edn). MIT Press: Cambridge, MA.

Forrester JW. 1971. *World Dynamics*. Wright-Allen Press: Cambridge, MA.

Forrester JW. 1994. System dynamics, systems thinking, and soft OR. *System Dynamics Review* **10**(2–3): 245–256.

Freeman LC, Borgatti SP, White DR. 1991. Centrality in valued graphs: a measure of betweenness based on network flow. *Social Networks* **13**: 141–154.

Kampmann CE. 2012. Feedback loop gains and system behavior (1996). *System Dynamics Review* **28**(4): 370–395.

Kampmann CE, Oliva R. 2009. Analytical methods for structural dominance analysis in system dynamics. In *Encyclopedia of Complexity and Systems Science*, Meyers RA (ed.). Springer: New York; 8948–8967.

Oliva R. 2004. Model structure analysis through graph theory: partition heuristics and feedback structure decomposition. *System Dynamics Review* **20**(4): 313–336.

Richardson GP. 1995. Loop polarity, loop dominance, and the concept of dominant polarity (1984). *System Dynamics Review* **11**(1): 67–88.

Schaffernicht MF, Groesser SN. 2014. The SEXTANT software: a tool for automating the comparative analysis of mental models of dynamic systems. *European Journal of Operational Research* **238**(2): 566–578.

Schoenenberger L, Schenker-Wicki A. 2014. Can system dynamics learn from social network analysis? In Proceedings of the 2014 International System Dynamics Conference. System Dynamics Society: Delft, Netherlands.

Sterman JD. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. McGraw-Hill Higher Education: Boston, MA.

DOI: 10.1002/sdr

Tarjan R. 1972. Depth-first Search and linear graph algorithms. *SIAM Journal on Computing* **1**(2): 146–160.
Thun J. 2006. Maintaining preventive maintenance and maintenance prevention: analysing the dynamic implications of Total Productive Maintenance. *System Dynamics Review* **22**(2): 163–179.
Wolstenholme EF. 2003. Towards the definition and use of a core set of archetypal structures in system dynamics. *System Dynamics Review* **19**(1): 7–26.
Wolstenholme EF. 2004. Using generic system archetypes to support thinking and modelling. *System Dynamics Review* **20**(4): 341–356.
Wunderlich P, Grössler A, Zimmermann N, Vennix JAM. 2014. Managerial influence on the diffusion of innovation within intra-organizational networks. *System Dynamics Review* **30**(3): 161–185.

Supporting information

Additional supporting information may be found in the online version of this article at the publisher's web-site.

Appendix

Table 1.A. Algorithmically detected underachievement archetypes from the SILS

Intended consequence (ic)				Unintended consequence (uc)		
Underachievement archetype						
No.	D	P	Loop	D	P	Loop
1*	3	R	1 → 15 → 1	3	B	1 → 2 → 1
2*	3	R	1 → 15 → 1	5	B	1 → 8 → 7 → 2 → 1
3*	3	R	1 → 15 → 1	7	B	1 → 8 → 23 → 17 → 6 → 2 → 1
4	3	R	1 → 15 → 1	7	B	1 → 31 → 30 → 28 → 5 → 2 → 1
5*	3	R	1 → 15 → 1	8	B	1 → 46 → 45 → 57 → 4 → 2 → 1
6*	3	R	1 → 15 → 1	10	B	1 → 46 → 45 → 57 → 18 → 17 → 6 → 2 → 1
7*	3	R	1 → 15 → 1	11	B	1 → 54 → 52 → 49 → 50 → 30 → 28 → 5 → 2 → 1
8*	7	R	1 → 31 → 30 → 28 → 14 → 15 → 1	3	B	1 → 2 → 1
9*	7	R	1 → 31 → 30 → 28 → 14 → 15 → 1	5	B	1 → 8 → 7 → 2 → 1
10*	7	R	1 → 31 → 30 → 28 → 14 → 15 → 1	7	B	1 → 8 → 23 → 17 → 6 → 2 → 1
11*	7	R	1 → 31 → 30 → 28 → 14 → 15 → 1	8	B	1 → 46 → 45 → 57 → 4 → 2 → 1
12*	7	R	1 → 31 → 30 → 28 → 14 → 15 → 1	10	B	1 → 46 → 45 → 57 → 18 → 17 → 6 → 2 → 1
13*	10	R	1 → 31 → 48 → 46 → 45 → 57 → 13 → 15 → 1	3	B	1 → 2 → 1
14*	10	R	1 → 31 → 48 → 46 → 45 → 57 → 13 → 15 → 1	5	B	1 → 8 → 7 → 2 → 1
15*	10	R	1 → 31 → 48 → 46 → 45 → 57 → 13 → 15 → 1	7	B	1 → 8 → 23 → 17 → 6 → 2 → 1
16*	10	R	1 → 31 → 48 → 46 → 45 → 57 → 13 → 15 → 1	11	B	1 → 54 → 52 → 49 → 50 → 30 → 28 → 5 → 2 → 1
17*	11	R	1 → 25 → 24 → 31 → 32 → 21 → 17 → 12 → 15 → 1	3	B	1 → 2 → 1
18*	11	R	1 → 25 → 24 → 31 → 32 → 21 → 17 → 12 → 15 → 1	5	B	1 → 8 → 7 → 2 → 1
19*	11	R	1 → 25 → 24 → 31 → 32 → 21 → 17 → 12 → 15 → 1	8	B	1 → 46 → 45 → 57 → 4 → 2 → 1
20*	11	R	1 → 25 → 24 → 31 → 32 → 21 → 17 → 12 → 15 → 1	11	B	1 → 54 → 52 → 49 → 50 → 30 → 28 → 5 → 2 → 1

D, delay; P, loop polarity; B, balancing; R, reinforcing; *MSILS.

DOI: 10.1002/sdr