A Domain-Specific Language to Process Causal Loop Diagrams with R

Adrian Stämpfli

Abstract Causal Loop Diagrams (CLDs) are a flexible and valuable tool for diagramming the feedback structure of systems. In strategic decision-making and management, we use CLDs to structure and explore complex decision-making situations, to foster learning, as a basis for simulation models, and to communicate simulation results. However, the crucial dissemination of CLDs and the possible learnings beyond the project-team is challenging.

To overcome this problem, we developed a Domain-Specific Language that allows modeling experts with little programming experience to generate visual representations of CLDs that (1) replace the most complicated CLD elements with a step-by-step explanation and (2) strive to lower the barriers to learning while addressing a broader target audience.

Keywords Strategic decision-making · Causal Loop Diagrams · System Dynamics · Domain-Specific Languages · R

1 Introduction

System Dynamics (SD) is a method for modeling and simulation of complex systems that adapts control theory to a broader set of problems [1]. Two key elements differentiate SD from other methods in Operations Research. (1) SD models *generate dynamics endogenously*. Many classical SD models show how flawed internal policies of industries or cities generate decay without external limiting factors [2]. (2) SD *makes mental models explicit* by modeling them as CLDs. Making the models explicit is the basis for a deeper understanding of a

A. Stämpfli (✉)
Institute of Modeling and Simulation, University of Applied Sciences St. Gallen, St. Gallen, Switzerland
e-mail: adrian.staempfli@fhsg.ch

J. S. Neufeld et al. (eds.), *Operations Research Proceedings 2019*, Operations Research Proceedings, https://doi.org/10.1007/978-3-030-48439-2_79

messy situation, for revising mental models, for allowing double-loop learning to occur, and for taking strategic decisions [3–6].

1.1 Causal Loop Diagrams

Causal Loop Diagrams (CLDs) are a widely used, flexible, and valuable tool for diagramming the endogenous perspective or feedback structure of systems [8, 9]. CLDs are used (1) to structure complex problems; (2) to explore complex decision-making situations in participatory modeling processes; (3) to foster learning among stakeholders involved in the modeling process; (4) as a basis for simulation models, and (5) to communicate results of simulation studies [4, 5, 10].

Mathematically, CLDs are a particular sort of directed cyclic graphs. They consist of variables (in graph terminology: *vertices*) that are connected by directed causal links (*edges*). Each link states a causal influence from the starting variable to the ending variable. Each causal link is assigned a polarity, either positive (+) or negative (-), that indicates how the dependent variable changes when the independent variable changes [9]. Each feedback loop (*cycle*) describes positive (reinforcing, *R*) or negative (balancing, *B*) feedback [9].

CLDs are an essential tool to foster learning and feedback processes among the stakeholders involved in a project [4, 10]. However, the crucial dissemination of those learnings beyond the project team is difficult and requires knowledge about CLDs that senior decision-makers generally do not have [10, 11]. This impedes the acceptance of CLDs and the implementation of project results [10].

2 An Embedded Domain-Specific Language to Process CLDs in R

To overcome this problem, we developed an embedded Domain-Specific Language (DSL) in R that allows us to import CLDs from standard SD tools into R and to generate plots of those CLDs.[1,2] The plots focus on specific aspects or parts of the CLDs. By lowering the 'technicalness' and improving visual attractiveness, we address a bigger audience. Existing research shows that the most complicated elements of CLDs are the link and loop polarity signs. We thus replaced those elements: The DSL allows explaining a CLD step-by-step by highlighting specific causal chains and adding textual descriptions. The whole CLD is shown greyed-out throughout the process, which ensures that the target audience receives the CLD

[1] At the moment we support Vensim. Support for Stella, the other major SD tool, is planned for a future release.

[2] The DSL source code and further instructions are hosted at https://github.com/ims-fhs/cld.

as a whole. We assume that CLDs explained with the DSL allow for the same insights as standard CLDs. Moreover, we assume that such CLDs improve the chances of implementing insights generated within SD projects, because they (1) help to broaden the possible audience of project results (because simplified visual representations fit better to decision-makers' mental models) [10] and, therefore, (2) help to maintain overlaps between participants mental models and project results [4, 11].

2.1 The R Language and Embedded DSLs

R is an open-source programming language and software environment designed for statistical computing, data science, and graphics [12]. R is a very flexible programming language. The combination of first-class environments, lexical scoping, non-standard evaluation, and meta-programming make R especially well suited to support the creation of embedded DSLs [13].

DSLs are computer programming languages of limited expressiveness focused on a particular domain [14]. Embedded DSLs are composed of valid code in the general-purpose language (GPL) they are written. They take advantage of that GPLs parsing and execution framework [13]. The GPL naturally defines the syntactical bounds for the embedded DSLs [13, 15].[3]

The usage of embedded DSLs offers some advantages over the usage of standard libraries or frameworks: (1) A sense of *fluency* and a *language nature* can be achieved by the usage of only a few design patterns [14]; (2) The DSL approach increases programmer productivity [14], and (3) Embedded DSLs allow to write code that is better understandable by domain experts. Domain experts, therefore, can challenge the code more thoroughly. Requirements refinement, presumably the most challenging part of software development, is improved.

2.2 DSL Grammar

Following, we present the developed DSL by discussing its grammar. The DSL allows us to (1) `'IMPORT'` CLDs from standard SD tools into R; (2) `'LINK'` CLD elements; (3) `'DESCRIBE'` groups of CLD elements with textual descriptions; (4) `'PLOT'` the resulting graphics. We specify the grammar by using a standard notation.[4]

[3] Relatively flexible languages like R let you bend the rules a little bit using non-standard evaluation and meta-programming [15].

[4] The notation is a simplified version (with only two operators) of the Backus-Naur form.

DSL Expressions and DSL Sentences Every sentence of the DSL starts with the verb `IMPORT`.[5] The verb `PLOT` is only allowed at the end of a sentence. The verbs `LINK` and `DESCRIBE` can be composed in any possible way. This leads to the following grammar:

```
    DSL_SENTENCE ::= DSL_EXPRESSION
                   | DSL_EXPRESSION %>% PLOT

  DSL_EXPRESSION ::= IMPORT
                   | DSL_EXPRESSION %>% LINK
                   | DSL_EXPRESSION %>% DESCRIBE
```

A valid `DSL_SENTENCE` can be any `DSL_EXPRESSION` with or without `PLOT` at the end. A valid `DSL_EXPRESSION` can be any combination of an `IMPORT` at the beginning and arbitrary chains of `LINK` and `DESCRIBE` actions afterward. We use the pipe operator (`%>%`) to connect the actions inside a sentence.[6]

Causal Chains and Link Expressions We further need a sub-grammar to specify causal chains inside the verb `LINK`. We need to consider regular causal chains as well as the two extreme cases of causal chains: single variables and loops. We implemented an infix operator [15] (`%->%`) called *link operator* to specify the causal chains. We also need to select more than one causal chain in one link expression. The following grammar covers all cases:

```
  LINK_EXPRESSION ::= CAUSAL_CHAIN
                    | CAUSAL_CHAIN , LINK_EXPRESSION

     CAUSAL_CHAIN ::= VARIABLE
                    | CAUSAL_CHAIN %->% VARIABLE
```

2.3 An Example

We use the same CLD as shown in Fig. 1, to demonstrate example usage of the developed DSL. Consider the following DSL statement:

```
cld %>%
    link(gap %->% actions, gap %->% pressure) %>%
    link(actions, pressure) %>%
    describe(type = "text", "A gap not only leads to actions
        towards closing the gap, but also to pressure to adjust
        the long time goals.") %>%
    plot()
```

[5] Import covers two cases here: (1) the case, where we import a CLD; (2) the case, where we select an already imported CLD.

[6] This is an application of the Expression Builder pattern in combination with the Method Chaining pattern [14] exploiting R's possibilities to define custom infix operators [15] and doing non-standard evaluation in combination with meta-programming [13, 15].

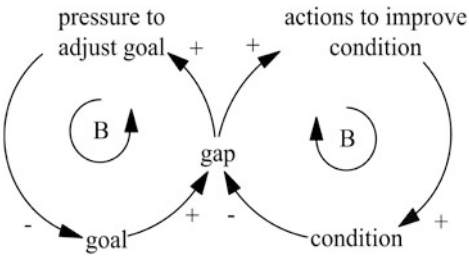

Fig. 1 A sample CLD that illustrates the *eroding goals* systems archetype [7], a situation where we provide actions to close a gap immediately, but at the same time accept that our goals decline over time

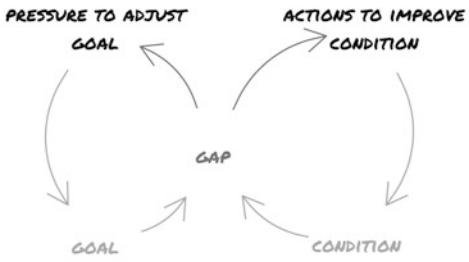

Fig. 2 One exemplary step in explaining the *eroding goals* CLD [7]

The first `link` statement highlights the variable gap and the two consequences of such a gap. The second `link` statement further highlights the two consequences. The `describe` statement adds a textual description. The resulting plot is shown in Fig. 2.[7]

3 Conclusions

CLDs are an essential tool to foster learning and feedback processes among the stakeholders involved in a project [4, 10]. The crucial dissemination of those learnings beyond the project team is, however, difficult and requires knowledge about CLDs that senior decision-makers generally do not have [10, 11]. To overcome this problem, we developed a DSL that allows generating visual representations of

[7]The colors are adjusted to grey tones for better printing results.

CLDs which replace the most complicated elements with a step-by-step explanation. In detail, this is solved using the following elements: (1) The *greyed out model with highlights* is an elegant way to study a CLD without concealing the circular structure at work. Having the feedback structure always there as a whole ensures that the target audience receives all the graphics as being part of a single CLD; (2) *Highlighting certain elements* helps to break the CLD into understandable pieces; (3) *Enriching the graphics with textual descriptions* allows emphasizing important mechanisms.

CLDs, we explain using the DSL expectedly allow for the same systems insights as the original CLDs do. In combination with the sketchy and handmade look and feel, we strive to improve the acceptance of CLDs for stakeholders of the 'untechnical' kind.

To implement the solution in the form of an embedded DSL in R proves valuable as well. Thanks to the DSL approach, we can write short, simple, and elegant code, which in turn provides for excellent prototyping possibilities. R's properties allowed us to find surprisingly simple notations, grammars, and suitable plotting possibilities.

In numerous client projects, the DSL turned out to be a very valuable tool: (1) to develop a common problem understanding; (2) to communicate that understanding to stakeholders beyond the project team; (3) to foster strategic decision-making. A particular appealing application of the developed DSL is a project funded by 'Innosuisse—Swiss Innovation Agency' in the field of policy design for elderly care.[8]

Future research is needed (1) to integrate the delay marks used in standard CLD notation and (2) to explore further possibilities that enhance the DSL expressiveness e.g., reference mode graphs or simulation capabilities.

References

1. Forrester, J.W.: Industrial Dynamics. M.I.T. Press, Cambridge (1961)
2. Richardson, G.P.: Reflections on the foundations of system dynamics. Syst. Dyn. Rev. **27**, 219 (2011)
3. Torres, J.P., Kunc, M., O'Brien, F.: Supporting strategy using system dynamics. Eur. J. Oper. Res. **260**, 1081–1094 (2017)
4. Lane, D.C.: Modelling as learning: a consultancy methodology for enhancing learning in management teams. Eur. J. Oper. Res. **59**, 64–84 (1992)
5. Vennix, J.A.M.: Group model-building: tackling messy problems. Syst. Dyn. Rev. **15**(4), 379–401 (1999)
6. Paich, M., Sterman, J.D.: Boom, bust, and failures to learn in experimental markets. Manag. Sci. **39**, 1439–1458 (1993)

[8]More information on the on-going project can be found at https://www.fhsg.ch/de/forschung-dienstleistungen/institute-zentren/institut-fuer-modellbildung-simulation/care-system-design/verbesserte-planung-der-langzeitpflege/.

7. Senge, P.M.: The Fifth Discipline: The Art and Practice of the Learning Organization. Doubleday/Currency, New York (1990)
8. Lane, D.C.: The emergence and use of diagramming in system dynamics: a critical account. Syst. Res. Behav. Sci. **25**, 3–23 (2008)
9. Sterman, J.: Business Dynamics: Systems Thinking and Modeling for a Complex World. Irwin/McGraw-Hill, New Delhi (2000)
10. Wolstenholme, E.F.: Qualitative vs quantitative modelling: the evolving balance. J. Oper. Res. Soc. **50**, 422 (1999)
11. Hovmand, P.S.: Community Based System Dynamics. Springer, New York (2014)
12. Ihaka, R., Gentleman, R.: R: A language for data analysis and graphics. J. Comput. Graph. Stat. **5**, 299 (1996)
13. Wickham, H.: Advanced R (CRC Press, Boca Raton, 2015)
14. Fowler, M.: Domain-Specific Languages (Addison-Wesley, Boston, 2011)
15. Mailund, T.: Domain-Specific Languages in R: Advanced Statistical Programming. Apress, New York (2018)