Tutorial: Discover Discovery Rules for your Modeling Language*

Ilia Bider[1,2]

[1] Stockholm University, Borgarfjordsgatan 12, Kista, Stockholm, 164 55, Sweden
[2] University of Tartu, Ülikooli 18, 50090 Tartu, Estonia

Abstract
The tutorial will introduce a new concept – discovery power – that can be used to characterize an enterprise modeling language. The concept is different from, but connected to, the concept of expressive power. The concept is defined as "the degree of help provided by the structure of an enterprise modeling language to expand a partly built model or fill gaps in it". The concept is realized by discovery rules that are different for different modeling languages.

Keywords
Modeling languages, Discovery power, Discovery rule [1]

1. Introduction

The concept of discovery power of a modeling language was introduced in our paper [1] at 13th Enterprise Design and Engineering Working Conference (EDEWC 2023) in Wien, in 2023. The concept is informally defined as the "Degree of help provided by the structure of modeling language to expand a partly built model or fill gaps in it." What the modeling language helps to discover depends on the language, as the discovery power differs for different languages. For practical purposes, the discovery power of a modeling language can be represented as a set of rules that guide the modeler on what else can be found and depicted based on what is already depicted in the model.

The important feature of the concept of discovery power is that it can be applied to any modeling language that has a diagrammatic presentation. Since the concept introduction, another conference paper at BIR 2024 was published [2]. The paper is devoted to discovery rules for depicting tacit knowledge usage and management in Fractal Enterprise Models (FEM) [3].

2. How it Works

To explain how the discovery rules works, we will use our own version of the OODA (Observe-Orient-Decide-Act) loop suggested by Boyd [4], the first two parts of which are depicted in Fig. 1, The central point of the OODA loop is *Orient* – an activity that allows one to get a clear view of the situation, which is then used to generate a hypothesis for action (*Decide*). The orientation in the *Orient* phase is done based on the information from the *Observe* phase obtained under the guidance and control from the *Orient* phase.

Feed forward from the *Observe* phase to the *Orient* phase is information based on which an enterprise model is being built. Feed forward from *Orient* to *Decide* is an enterprise model based on which a decision on actions is taken. In general, the model should include information about the

* *ER 2025: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos, October 20-23, 2025, Poitiers, France*
1* Corresponding author.
ilia@dsv.su.se (I. Bider)
0000-0002-3490-6092 (I. Bider)

environment of the enterprise in focus, e.g., competitors, and about the enterprise itself, e.g., its capabilities. The information needed to build the model can be obtained from different sources, including documents describing the environment or the organization itself, interviews with stakeholders, facilitating workshops with them, and actual observations of what is happening outside or inside the organization.

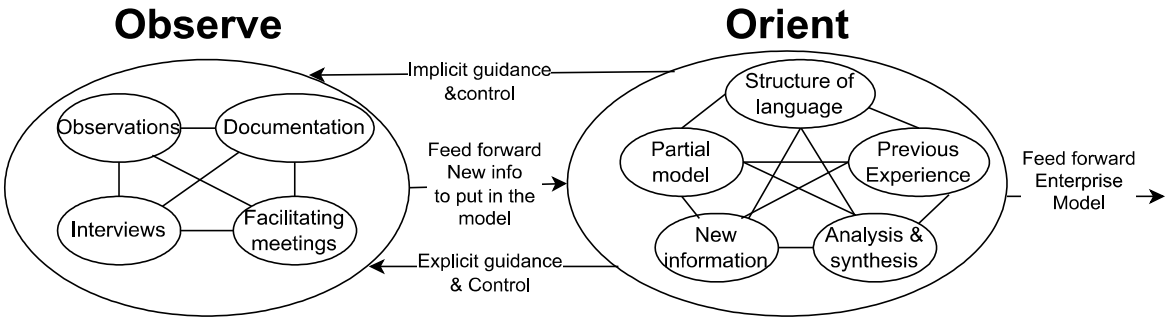

Figure 1: Adaptation of OODA cycle for a case when orient should produce an explicit model, only two phases presented (adapted from [1]).

Gathering new information in the Observe phase is guided and controlled partly implicitly partly explicitly. Explicit guidance and control can be in the form of questions for interviews, instructions on what specific information is missing and needs to be obtained, etc. This control is partly defined by the experience and background of the modelers, which is depicted in Fig. 1 as *Previous experience*, *Cultural Heritage* and *Genetic Heritage*. The other part is provided by the structure of the modeling language that is used to build the model. This structure defines which entities and relations should be found in the environment and organization. For example, based on this structure, specific interview questions can be asked, or gaps in the model being built initiate looking for certain kinds of information in the documents.

3. Examples of Discovery Rules

Paper [1] suggests a way to operationalize the discovery power of a language so that it can be used in practice. It is done by having a set of rules that show what can be added to the model in case some specific elements already exist in it. We will be using the rule format suggested in paper [2]. This format presumes that the language elements consist of shapes and connections between them, which are shown by lines of a certain form. An example of such a rule for the ArchiMate language [5] is presented in Fig. 2. This rule consists of two parts. One that shows what is already

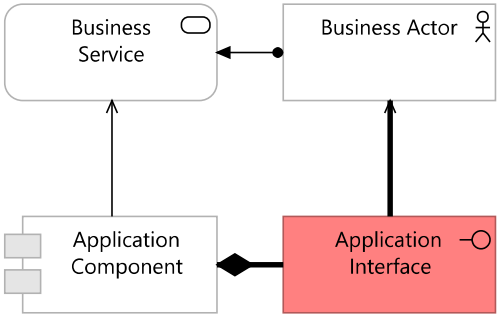

Figure 2: An example of a rule for ArchiMate language.

in the model, and one that shows what could be added to the model based on additional investigation. The elements that are added are highlighted. Added shapes have a rose background, while added connections are highlighted by having thicker lines. The rule presented in Fig. 2 means that for the actor of a business service supported by an application component, there, most probably, should exist an interface via which this actor can work with the application component.

Note, however, that this needs to be established for each actor separately, because some actors may not use this application component. Also, two different actors can use the same interface. It can be a sign that the system is not properly designed.

If the tool for drawing models does not have a feature that allows for having thicker relations, the two-part rule definition can be used, as shown in Fig. 3, which expresses the same rule as in Fig. 2.

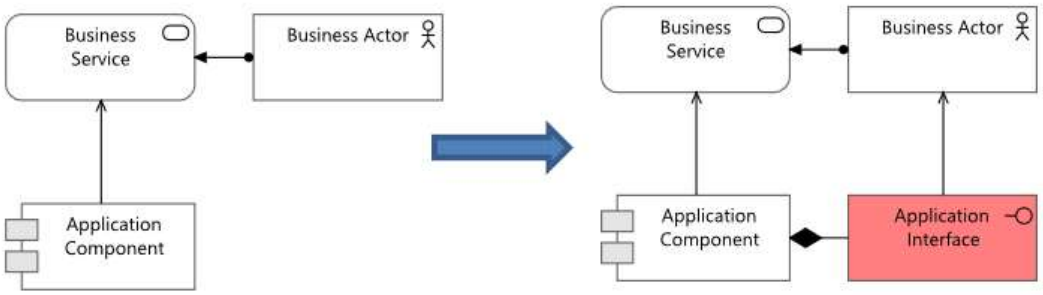

Figure 3: Another way of depicting discovery rule**s.**

In Fig. 4, we present another discovery rule, this time for Fractal Enterprise Model (FEM) [3]. It uses the same format as in Fig. 1. It shows that for the recruiting process, there needs to be some attraction that will appeal to people to become employees. It can be a salary, a modern building in the center of the town, etc. The rule says that if you are recruiting people for a position in your company, you need to have something that attracts them.

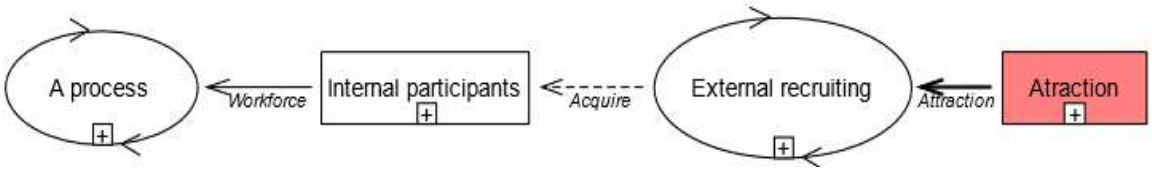

Figure 4: A discovery rule for FEM.

In Fig. 5, we present an example of a discovery rule for IDEF0 [6]. Suppose the output of a functional element has been established. In that case, the next step is to find where it goes, thus finding another functional element that consumes the output as input, control, or mechanism. This rule is presented in the same way as the rule in Fig. 3, i.e, as consisting of two parts.

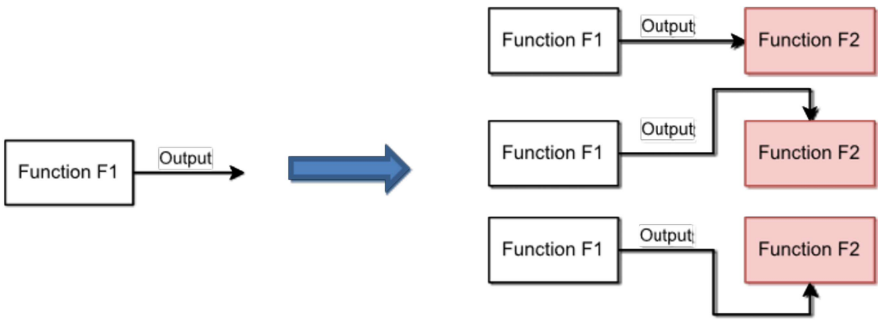

Figure 5: A discovery rule for IDEF0

Note that the labels in the shapes are abstract; they do not need to be matched by the labels in the actual, partly built model when the rule is applied. The matching should be done based on the shapes of the elements and connections between them, including labels on the connections.

4. Objective of the Tutorial

The objective is that the attendees will learn to use the concept of *discovery power* by working on discovery rules for a modeling language of their choice. The intention is that the attendees will continue working on discovery rules for their preferred language and will use them in their teaching or modeling practice. One possibility is that the tutorial results in a common research paper with examples of discovery rules for different modeling languages.

5. Tutorial plan

The first part (30 min) will be the introduction to the concept of discovery power, and how it can be practically realized in a number of discovery rules. In the second part (30 min), the participants will form groups according to the preferred modeling language, and they will create some discovery rules for this language. The tutorial presenter will go around groups to answer questions and, if needed, provide help. The tutorial will be finished by each group presents at least one discovery rule for their modeling language.

Acknowledgments. The work of the author was partly supported by the Estonian Research Council (grant PRG1226).

Declaration on Generative AI

The author has not employed any Generative AI tools.

References

[1] Bider, I., Perjons, E.: On the Concept of Discovery Power of Enterprise Modeling Lan-guages and its Relation to their Expressive Power. In Malinova Mandelburger, M., Guer-reiro, S., eds. : Advances in Enterprise Engineering XVII. EDEWC 2023, LNBIP, vol. 510, Springer (2024) pp. 92-106.

[2] Bider, I., Perjons, E.: Discovery Rules for Depicting Tacit Knowledge Usage and Manage-ment in Fractal Enterprise Models. In: Řepa, V., Matulevičius, R., Laurenzi, E: (eds.) Per-spectives in Business Informatics Research, LNBIP, vol. 529 (2024), pp. 209- 224. Springer.

[3] Bider, I., Perjons, E., Klyukina, V.: Tool Support for Fractal Enterprise Modeling. In : Do-main-Specific Conceptual Modeling. Springer (2022) 205–229.

[4] Boyd, J.: The Essence of Winning and Losing. In: Unpublished lecture notes, 12(23), pp.123-125. (Accessed May 2025) Available at: http://www.expertsecho.com/johnboyd/johnboyd_docs/Essence_of_Winning_and_Losing.pdf

[5] The open group: ArchiMate® 3.2 Specification. In: The Open Group. Available at: https://publications.opengroup.org/standards/archimate/specifications/c226, (Accessed 2025).

[6] 10. NIST: Integration definition for function modeling (IDEF0), Draft Federal Information Processing Standards, Publication 183, 1993. In: IDEF. Available at: https://www.idef.com/idefo-function_modeling_method/, (Accessed 2025) .