Towards Executable Business Processes with the Problem Oriented Engineering Process Algebra

Dariusz Wojciech Kaminski
Computing Department
The Open University, UK
dariusz.kaminski@gmail.com

Jon G. Hall
Computing Department
The Open University, UK
J.G.Hall@open.ac.uk

Lucia Rapanotti
Computing Department
The Open University, UK
L.Rapanotti@open.ac.uk

***Abstract*—The paper introduces a process algebra for business process models. The algebra is located within Problem Oriented Engineering, a framework for engineering design, and is based on a process pattern defined by Hall and Rapanotti by which Problem Oriented Engineering developments should be structured. The pattern is a generator for processes being composable in three ways: in sequence, in parallel and fractally. In explicating this process algebra, a machine readable and animatable CSP is used, which forms a semantic basis for the behaviour modelling of processes. The benefits of this algebra are: simplicity, support for business process analysis and synthesis, and explicit recognition of choices (and their impact) made by agents.**

***Keywords-process modelling; process algebra; Problem Orientation*.**

I. Introduction

Business process analysis and synthesis are greatly facilitated by executable models, such as Business Process Execution Language (BPEL), which can be represented graphically by Business Process Modeling and Notation (BPMN) [18]. Such approaches provide only the building blocks by which processes can be built, without any predictive capability of the properties of processes themselves.

A predictive model is, essentially, one with which "What if?" questions can be answered. An example question might be:

> What if we asked a stake-holder at this point whether we have understood their problem well enough; would the developmental risk we face change?

Being able to answer such questions may allow us to distinguish from all those that solve a business problem, those business processes best match other criteria such as, for instance, the availability of resources or our attitude to risk.

In this paper, we provide a model that we have developed in the context of Problem Oriented Engineering (POE) [6], [7]. POE has been shown to have predictive capability in the design of artefacts, and we hope, with this paper, to begin working towards using POE's predictive capabilities for business process design.

The paper is structured as follow. Section II provides a brief overview of business process modelling and Problem Oriented Engineering. Section III introduces and explains the POE Process Algebra, with an example in Section IV. Section V reflects on the contribution and its potential application.

II. Background and related work

POE is motivated by a view of engineering as a problem solving activity [15]. POE provides many tools for solving problems, including the POE Process Pattern (PPP) [5], upon which the treatment of this paper is based. Simply put, the PPP orders a problem solving activity as iterations between exploring the problem and exploring the solution, with interleaved validation activities, whilst recognising that both problem and solution exploration can also be seen as problem solving activities.

Under the POE view, business processes are designed artefacts that solve business problems. We assume that for any business problem there will be many possible candidate business process solutions, each with different characteristics – here we consider cost (or resource use), and risk – in the solution space. This paper provides the business process designer with tools that allow the visualisation of business processes along a 'risk/resource continuum', each of which can then be compared to an organisation's available resource and risk appetite, as illustrated in Figure 1.

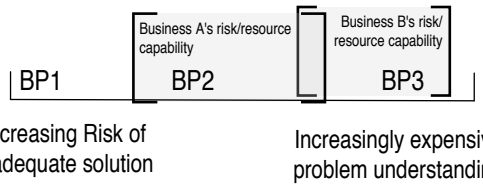

Figure 1. Two businesses A and B, each with different risk appetites and available resources, experience the same problem for which three candidate solutions – BP1, BP2 and BP3 – exist. Plotting each business process along the risk/resource continuum allows the most appropriate to be chosen.

Thus, whilst consistent with Aguilar-Savén's definition [2]: "a business process is the combination of a set of activities

 ISBN: 978-1-61208-165-6

within an enterprise with a structure describing their logical order and dependence whose objective is to produce a desired result", we provide hooks for the management of the following risk/resource trade-off [5]: although understanding the problem can be resource intensive in terms of developmental and stake-holder time, it can mitigate the risk of solving the wrong problem. For the purposes of this paper, we call this the "risk/resource trade-off assumption".

A. Business process modelling

Many analytical frameworks have been considered for the classification of business process models. Wang *et al.*, for instance, [17] surveys BPEL4WS (BPEL for Web Services), BPMN, UML (Unified Modeling Language), XPDL (XML Process Definition Language), Petri Nets, IDEF0, and IDEF3.

Given their easy representation of concurrency and their explicit representation of state, Petri Nets have long been associated with the modelling of Business Processes, including the work of Van der Aalst [1], and that based on the Petri Box model of Best et al. Our model shares with Petri Nets a formal model in POE, but is focussed on on completeness of representation only on the accurate representation of problem solving steps that exist within a business process. Moreover, we aim for synthesis of business processes not just their modelling for animation.

Aguilar-Savén provides a comprehensive review of a number of business process modelling techniques and tools, and also proposes a framework to classify the techniques/models according to their purpose. The classification proposed in [2] is done according to the purposes (descriptive, analytical, enactable), and change model permissiveness (passive or active). The approach proposed in this paper aims to be active in regards to permissiveness, and depending on the context and modelling goals the purpose falls into all four categories.

Other classification emphasising the link between modelling, decision and planning capabilities, but also their relation to entities (time, resources, causality and authority), was provided by Macintosh [11].

Mentzas et al. [13] provide an evaluation of alternative approaches to business process modelling with workflows. One of the problems, as cited by Mentzas et al., is the hardship in modelling exceptional tasks or processes, and their advice is to exclude such processes from process models, "due to uncertainty either in time or in the processing entities involved".

Perhaps unsurprisingly, synthetic approaches to business processes are fewer in number: A formal definition of structured workflow in terms of activities was provided by Kiepuszewski et al. in [9], where it was shown how these can be composed to form arbitrary workflow models, but require the use of powerful verification engines to help detect whether a composed process is well-behaved.

B. Problem Oriented Engineering

Problem Oriented Engineering is a framework for engineering design, similar in intent to Gentzen's Natural Deduction [16], presented as a sequent calculus. As such, POE supports rather than guides its user as to the particular sequence of design steps that will be used; the user choosing the sequence of steps that they deem most appropriate to the context of application. The basis of POE is the *problem* for representing *design problems* requiring designed solutions. *Problem transformations* transform problems into others in ways that preserve solutions (in a sense that will become clear). When we have managed to transform a problem to axioms[1] we have solved the problem, and we will have a designed solution for our efforts.

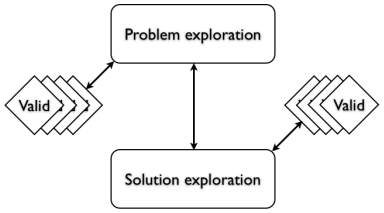

Figure 2. The POE Process Pattern: iteration between problem and solution exploration with interleaved validation (adapted from [5]).

A comprehensive presentation of POE is beyond the scope of this paper (but can be found in [6], [7]). For this paper it will be sufficient to consider the structure that POE suggests for problems solving steps that is illustrated in Figure 2 in which rectangles are resource consuming activities; diamonds indicate requests to stake-holders for validation either – on the left – of problem understanding, or – on the right – of a candidate solution.

The potential for looping in the POE process pattern concerns unsuccessful attempts to validate: unvalidated problem understanding will require problem rework as will an unvalidated solution. In this way, validation within the process has an impact on both (developmental) resources and risk: resource will vary *with* validation instances; risk varies inversely with validation instances.

C. Complex problem solving

Although the POE process pattern provides a structure for problem solving, in its raw form, a problem will only be solved when, after iteration, a validated problem is provided with a validated solution. This 'bang-bang' approach is suitable for simple problems, but is unlikely to form the basis of any realistically complex problem encountered in software engineering.

[1] An *axiomatic problem* is a problem whose adequate, i.e., fit-for-purpose, solution is already known.

 ISBN: 978-1-61208-165-6

To add the necessary complexity, the POE process pattern combines with itself in three basic ways; in combination, it is again a process that can be combined. The three ways it can be combined are in sequence, in parallel and in a fractal-like manner, as suggested in Figure 3, and as described in the sequel.

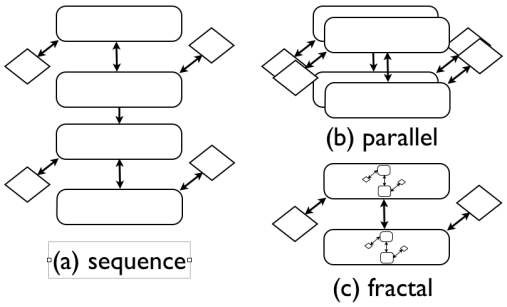

Figure 3. Problem solving can be performed (a) in sequence, and (b) in parallel. Under POE, problem and solution exploration are problems solving activities, so that *fractal* composition is also possible.

Briefly, in sequence, the POE process pattern models (more or less traditional) design processes in which existing structures, such as architectures, are used as structure in the solution space according to significant requirement and qualities, and according to developmental requirements.

If resources exist, parallel problem solving is possible. Of course, communications between those involved in parallel development is a non-trivial issue; we do not consider it here, though.

Fractal-like Design: [5] explains in detail how problem and solution exploration can be seen as problem solving processes: although we do not go into detail, essentially, the problem to be solved by problem exploration is to find the problem (or solution) description that satisfies the validating stake-holder. This allows us to embed within problem and solution exploration copies of the PPP, making the process self-similar in the sense that the whole structure resembles the parts it is made of.

Trusted processes: A POE process is *trusted* when there are no validating stake-holders. As argued in [5], trusted processes exist to 'bottom out' problem solving; given that no validator is involved they do not have a fractal form.

D. Validation

We wish to model the validation relationships that occur in organisations. In POE, we assume that there are two types of validation relationship: (i) *direct validation*, in which one actor determines whether the problem to solve has been understood or the solution is adequate; (ii) *delegated validation*, in which one actor – the delegator – delegates to one (or more) actors who then can validate either directly or through delegation. Of course, delegation does not transfer the responsibility for the outcomes of a choice, so that at a future point the delegator may check that the choice has been made, and also that the outcome and justification of that choice is suitable.

A delegator can clearly not usefully delegate to themselves, nor can there be cycles in the delegator/delegatee relationship. The reflexive transitive closure of the relationship is, therefore, a partial order.

Thus, POE see validation as a *social choice* [3], which can be expressed as a *partially ordered set*, or *poset*, i.e., $(X, <)$, where X is the domain set and symbol $<$ is the delegation relation, which we call the *validation structure*, or *VStruct*.

Example 1: Consider three validators – v_1, v_2 and v_3 – with v_i delegates to v_j, $v_i < v_j$, whenever $i > j$. The validation structure is the poset $(V, <)$ with:

$$V = \{v_1, v_2, v_3\}, \qquad (v_1 < v_2), (v_2 < v_3)$$

We now turn to the modelling of business processes in POE.

III. POE Process Algebra

The aim is to define a process algebra to describe POE processes in terms of simple operations. These operations should allow for encoding and modelling of distinct exploration and validation activities. Firstly, they should provide compositions for the three basic ways in which POE processes combine. Secondly, the operations should be sufficient to express arbitrarily complex process models, whose structure and behaviour could be modelled and reasoned about.

To this end, we define the POE Process Algebra (PPA).

A process under PPA is formed under the following syntax:

$$P = \beta \mid P; P \mid P \| P \mid P \; {}_V\!\bowtie_V \; P$$

in which a basic POE Process, β, combines to produce sequence $P; P$, parallel $P \| P$, and fractal processes $P_V \bowtie_V P$, the latter in combination with validation structures[2].

The operators correspond to the possibilities for combination of POE Processes as described in Section II.

A. Executable semantics of PPA

We provide an executable semantics for PPA terms using machine readable CSP (CSP-M) [4], as implemented in *ProB* tool [10]. ProB was chosen as it provides for the animation and model-checking of the resulting CSP models and traces [8] on the trusted processes. A CSP semantics allows us to reason about processes in terms of the effects of basic process on resource, and on their relationship to validation, through fractal composition.

[2]The astute reader will note the lack of a choice composition, often included in process algebras that express computation. This is because the PPP describes the structure of the problem solving process and not the (creative) choices that are expressed therein: POE makes no comment on the creativity of the design process, that is contextually determined by the stake-holders whose notions of adequate apply.

We have already been able to use *ProB*'s model checking and animation functionality to test whether POE processes (encoded as POE Programs) execute and complete as expected. Examples follow.

B. POE Programs language syntax

We designed a language for encoding POE processes using the operators from PPA, and as a function over POE Programs in this language we created a set of tools to translate PPA encoded input to CSP-M.

The executable model for POE processes was implemented in the Ruby programming language [12]. The input POE Program is translated to CSP-M output, and this in turn can be directly used in *ProB* tool. Further technical details of the implementation are beyond the scope of this paper.

C. A CSP-M semantics of PPA

Our semantics is over the domain of CSP-M expressions and so we must associate with each POE Process expression a CSP-M term. Most choices for the semantics are made simple due to the process algebraic nature of the source and target languages: CSP is an algebra as is PPA. Below, we describe in detail only the more difficult encodings.

PPA trusted in CSP: Trusted POE processes are the building block from which others are built. Their defining characteristic is that they make no use of validators. As such they can be, essentially, any piece of CSP-M code.

Sequence: Sequential composition under PPA is, as might be expected, implemented using CSP's sequential operator ;. Simply:

```
Sequence(Left,Right) = Left ; Right
```

Parallel: Although there are other choices, in this initial semantics parallel composition under PPA is implemented using CSP's interleaving operator. Other choices would allow the processes involved to communicate with each other to, for instance, model the passing of documentation between them. Such details are left for a fuller description.

```
Parallel(Left,Right) = Left ||| Right
```

PPA fractal in CSP: Referring to Figure 3, fractal composition in our CSP-M implementation must accept four arguments: $Upper$ for the problem exploration process, $Lower$ for solution exploration process, and the two respective validation structures, which we will call *VSPID* (for *V*alidation *S*truct for *P*roblem) and VSSID (for *V*alidation *S*truct for *S*olution). Our semantics simply places the CSP-M semantics of the operands together through CSP-M's interleaving operator.

```
Fractal(Upper,Lower,VSPID,VSSID) = (Upper |||
    ValidateUpper(VSPID) |||  Lower |||
       ValidateLower(VSSID))
```

Table I
SUMMARY OF KEY TERMS

Term	Description
Mortgage Servicing	Managing mortgage loans – interest accruals, billing, collecting due payments, redemption of loans, etc.
Financial Services Authority (FSA)	Regulatory body for financial institutions
(Mortgage) Servicing Software	Software used in servicing activities
Triage Document	A form used to report details of a production issue

IV. CASE STUDY EXAMPLE

As an example of the application of our algebra, we model the process described in [14], from which the following description is adapted.

The context of the study is a UK based subsidiary of an American financial organisation (the Company), with business, systems and technical analysts based in the UK, technical architects in the US and development staff in India. Relevant key terms and stake-holders are summarised in Tables I and II. The company supplies Mortgage Servicing Software package to its Client, a product that manages loan accounts once mortgage payments have been made by the Client's customers. The software facilitates business tasks such as payment calculation and processing, account queries, early redemption, correspondence, interest rate change and customer billing. The company also provides support and assists in the resolution of issues that arise during the use of the supplied software.

Recently, the company lost a number of subject matter experts but retains a contractual obligation to provide support to the Client to enhance and maintain the supplied software stack. This motivated the company to investigate through this study the capture of design rationale during Client's issue resolution. As many organisations face such losses, the success of our case study takes on increased importance. The case study also provided opportunities to consider how the application of POE techniques could improve the current issue resolution process. Process improvement is also an issue faced by many organisations.

A. Issue Resolution Process

When an issue is found in the Client's use of the Company's applications, a *Triage Document* is raised to describe the problem with information included that may assist in tracking down its cause. The reported issue is given a priority by the Client (*low*, *medium*, *high*) that governs the timeline for response and solutions, based on service-level agreements. Once the Triage Document is received by the Supplier's Production Support Team, an incident number is generated

 ISBN: 978-1-61208-165-6

Table II
SUMMARY OF KEY STAKE-HOLDERS

Stake-holder	Description
Client/Financial Institution	The mortgage institution managing customer mortgages
Customers	Patrons of the client organisation whose mortgages are being managed
(Servicing) Software Company	Company that supplies and maintains mortgage servicing software for the client. The case study is based in this organisation
Client Production Support Team	Client-side team tasked with resolving application software issues. Understand the workings of the application system and its platform, provide initial information on issues and communicate with business decision makers when questions arise of a business policy nature
Supplier Production Support Team (SPST)	Supplier-side team tasked with resolving issues with application software. In contact with CPST, assign work to the development and manage releases of solutions to the Client
Application Architect (AA)	Reviews a solution to assess if solution complies with standards
Product Assurance Team	Ensures the quality of the provided solution
Mort. Bus. Man.	Make final sign-off decision
IT Management	Make decision whether to implement solution in production
(Offshore) Development Team (DT)	Group of individual on the Supplier side tasked with developing software

and used to track the issue. The information is checked to see if it is sufficient for the investigation to progress.

Further discussions may be held between the Client and Supplier Production Support Teams to agree a) which issues lie with the application software and b) an approach for dealing with the issue. Additional clarification may be sought from the Client from which the issue report originated. The clarification may be in the form of screen shots of the application error, data extracts, event logs and example scenarios. When issues are agreed between CPST and SPST, they are analysed and solution approaches proposed by the development and architecture resources assigned to the issue. The proposed solutions are discussed with the CPST. Once agreement is reached on a solution approach, it is developed and tested. On completion of development and testing, the solution is packaged by the SPST with release notes and a test report, and delivered to the CPST. Subsequently, the CPST validate the delivered package, perform some further tests in collaboration with the Client, and may either return it for rework if it is unsatisfactory or implement it to the production systems if satisfied with the results.

B. The PPA Model

From this description, we identify four trusted processes, `Report`, `Raise`, `Analyse` and `Deliver`, and three fractal instances of Figure 2, here named `F1`, `F2` and `F3`, with associated validation structures `U`; `V1` and `V2`; `W1` and `W2`. We note that only the problem exploration part of `F1` is validated. The process is illustrated in Figure 4.

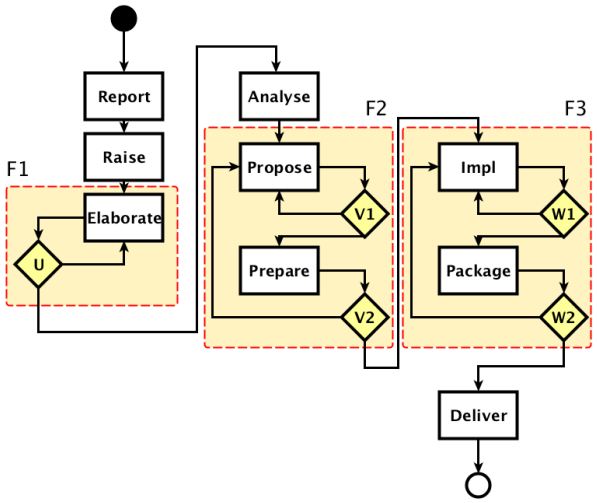

Figure 4. Processes; adapted from [14]

```
LET HiddenPV := (HiddenValidator)
LET U := (SPST1<CPST1<Client1)
LET V1 := (SPST2<AA1)
LET V2 := (CPST2<Client2)
LET W1 := (PA1)
LET W2 := (Client3<CPST3)

F1 := (ElaborateP{HiddenPV}><{U}ElaborateS)
F2 := (Propose{V1}><{V2}Prepare)
F3 := (Impl{W1}><{W2}Package)
P := Report;Raise;F1;Analyse;F2;F3;Deliver
```

Our tool generates approximately 180 lines of CSP-M to model the case study process of Figure 4. The part of the process corresponding to the level of that figure is:

```
P = Sequence(Sequence(Sequence(Sequence
      (Sequence(Sequence(T(pReport),
      T(pRaise)), Fractal(T(pElaborateP),
      T(pElaborateS), vsHiddenPV,vsU)),
      T(pAnalyse)), Fractal(T(pPropose),
      T(pPrepare), vsV1, vsV2)),
      Fractal(T(pImpl), T(pPackage),
      vsW1,vsW2)), T(pDeliver))
```

V. CONCLUSIONS AND FUTURE WORK

We have described our current model of Hall and Rapanotti's POE Process Pattern. The current model includes initial executable semantics of all aspects of the process, including fractal composition from which call-outs to validating stakeholders take place. Our model is encoded in a new process algebra, the POE Process Algebra (PPA), introduced here, and we have defined a semantic function over the algebra, which maps from CSP-M model that can be analysed in the ProB tool.

We have illustrated our PPA encoding on a business process that has appeared in the literature, presented a partial

 ISBN: 978-1-61208-165-6

CSP-M semantics of it as well as the ProB output of a partial exploration of its state space.

Our approach contributes to business process modelling by explicit recognition of choices that can be made by human agents, and with a devised language for representing POE Programs in the form of CSP-M, this approach provides syntax and semantics for behaviour modelling of business and validation processes in general.

Work that remains will consider how the executable model can be used to calculate resource usage of the process, and how risk and resources trade-off under, what we have termed, the risk/resource trade-off assumption of Section II. For, with such calculations, comes the possibility of systematic business process design in POE.

REFERENCES

[1] W.M.P. van der Aalst, "Making Work Flow: On the Application of Petri Nets to Business Process Management." LNCS, J. Esparza and C. Lakos, Eds. Springer, 2002, vol. 2360, pp. 1–22.

[2] R.S. Aguilar-Savén, "Business process modelling: Review and framework," *International Journal of Production Economics*, vol. 90, no. 2, pp. 129 – 149, 2004.

[3] G. Brightwell and D. B. West, "Chapter 11: Partially ordered sets," in *Handbook of Discrete and Combinatorial Mathematics*, K. H. Rosen, J. G. Michaels, J. L. Gross, J. W. Grossman, and D. R. Shier, Eds. CRC Press, 2000.

[4] M. Butler and M. Leuschel, "Combining CSP and B for Specification and Property Verification," in *FM 2005: Formal Methods International Symposium of Formal Methods Europe, Newcastle, UK, July 18-22, 2005. Proceedings*, LNCS, vol. 3582. Springer, 2005, pp. 221–236.

[5] J.G. Hall and L. Rapanotti, "Assurance-driven design in Problem Oriented Engineering," *International Journal On Advances in Systems and Measurements*, vol. 2, no. 1, pp. 119–130, 2009.

[6] J.G. Hall, L. Rapanotti, and M. Jackson, "Problem oriented software engineering: A design-theoretic framework for software engineering," in *Proceedings of 5th IEEE International Conference on Software Engineering and Formal Methods*. IEEE Computer Society Press, 2007, pp. 15–24.

[7] J.G. Hall, L. Rapanotti, and M. Jackson, "Problem-oriented software engineering: solving the package router control problem," *IEEE Trans. Software Eng.*, 2008.

[8] C.A.R. Hoare, *Communicating Sequential Processes*, ser. Series in Computer Science. Prentice-Hall International, 1985.

[9] B. Kiepuszewski, A.H.M. ter Hofstede, and C. Bussler, "On structured workflow modelling." in *CAiSE*, LNCS, B. Wangler and L. Bergman, Eds. Springer, 2000, vol. 1789, pp. 431–445.

[10] M. Leuschel and M. Fontaine, "Probing the Depths of CSP-M: A New FDR-Compliant Validation Tool," in *ICFEM '08: Proceedings of the 10th International Conference on Formal Methods and Software Engineering, Kitakyushu-City, Japan*. Springer-Verlag, 2008, pp. 278–297.

[11] A. Macintosh, "The need for enriched knowledge representation for enterprise modelling," in *AI (Artificial Intelligence) in Enterprise Modelling, IEE Colloquium on (Digest No.078)*, 7 1993, pp. 3/1 –3/3.

[12] J. McAnally and A. Arkin, *Ruby in practice*. Manning Publications Co. Greenwich, CT, USA, 2008.

[13] G. Mentzas, C. Halaris, and S. Kavadias, "Modelling business processes with workflow systems: an evaluation of alternative approaches," *International Journal of Information Management*, vol. 21, no. 2, pp. 123 – 135, 2001.

[14] A. Nkwocha, J.G. Hall, and L. Rapanotti, "Design rationale capture in the globalised enterprise: An industrial study," in *Proceedings of Fifth International Conference on Software Engineering Advances (ICSEA 2010)*. IEEE, 2010, electronic proceedings.

[15] G.F.C. Rogers, *The Nature of Engineering: A Philosophy of Technology*. Palgrave Macmillan, 1983.

[16] M.E. Szabo, Ed., *Gentzen, G.: The Collected Papers of Gerhard Gentzen*. Amsterdam, Netherlands: North-Holland, 1969.

[17] W. Wang, H. Ding, J. Dong, and C. Ren, "A comparison of business process modeling methods," in *Service Operations and Logistics, and Informatics, 2006. SOLI '06. IEEE International Conference on*, 21-23 2006, pp. 1136–1141.

[18] S.A. White, "Introduction to BPMN," *IBM Corporation*, May 2004.

 ISBN: 978-1-61208-165-6