

Preface

1. About this book

There are four levels of competence in Object Oriented Analysis (OOA):

1. Unconscious incompetence - you don't even know what OOA is
2. Conscious incompetence - you know what OOA is, but you don't know how to do it
3. Conscious competence - you know what OOA is and you can do it to some extent, but you still have to think about it
4. Unconscious competence - you are a true expert. You know what OOA is and you can do it without really thinking about it at all.

It's important for you to understand that by "incompetence" we simply mean "lack of competence", rather than "bad at". We're not using the term in a derogatory sense at all.

We (Ila and Jim) are fascinated by unconscious competence! This is the realm of the true expert, of mastery in a particular field of endeavor. Usually this expertise is acquired over many years, often through hard experience. But suppose we could model what an expert does, how they behave, what they think, how they approach their world (their map of the world)? Surely then we could teach this model to others so that they too can become experts. This is one of the key principles of Neuro Linguistic Programming - what one person can do, so can another. This is exactly what we try to do here.

Our original idea for this book was a simple volume of worked examples to be sold as a companion to our book, "UML 2 and the Unified Process" [Arlow 1]. We've had lots of requests over the years for a really complete worked example. However, as we began to write we became very aware that throughout the construction of the examples we were using particular thought processes and techniques that had been hard learned over many years of analysis and design. Nowhere (to our knowledge) had these thought processes and techniques been made explicit. In fact, until we had the leisure to work on these examples relatively free of time constraints, we had not even been consciously aware that we were using them. This is why we call them "secrets".

Other OOA books, such as our own "UML 2 and the Unified Process" tell the novice *what* to do *when*, and are very valuable and relevant. In fact, they are an essential prerequisite for a book such as this. However, they tend not to cover the *how* of OOA that experienced modelers apply often intuitively and unconsciously to get great results. After many years teaching OOA we observe again and again that beginners to analysis and design, even if told exactly *what* to do and *when* to do it, generally don't get the same quality of results as experienced analysts. Something is missing - the *how* that only the experts have. Thus a larger idea for this book was born.

We decided to model what we, as experienced analyst/designers, do and *how* we do it. Our goal is to impart this knowledge to you, the reader, in as efficient and entertaining a way as possible. We call the book "Secrets of Analysis" because the techniques we present here have been, until now, self-secret. That is to say no one has been consciously hiding them, it is just that the inner process of analysis, the *how* part, has never before (to our knowledge) been successfully modelled and presented in a book such as this.

We proceeded towards our goal through a process of modelling. In particular, Jim, being trained in Neuro Linguistic Programming (NLP) and in Ericsonian hypnosis, was already quite familiar with modelling success strategies. As we created the UML models in this book,

we also created informal conceptual models of *how* we were doing the analysis and design activities. These models were a revelation to us! When we stopped to examine them, we found we were doing things that were crucial to achieving success, but which we couldn't find written down anywhere. In fact, it has taken us many years to become sufficiently good modelers (in both the NLP and UML senses) to be able to model our own modelling strategies in a compelling way. Although these models might be interesting from an academic perspective, it is really the *insights* and *techniques* (secrets) that we discovered that are the real value as far as you, the practitioner, is concerned. It is these secrets that we present to you in this volume.

Our approach to OOA is pretty unique, and so it's appropriate to give it a name so that we can refer to it easily and so that you can tell your boss what you are doing. It also helps to clearly distinguish this inner, *how* based approach to OOA from more conventional *what*, *when* approaches. We call our approach Generative Analysis. It is generative because its focus is on generating information - how you capture information and how you transform information into new, more specific forms of information that take you ever-nearer to your goal. It is analytical not so much in the specific sense of OO Analysis but rather in the general sense in that it uncovers basic principles.

Through our studies in human communication we have learned many unique secrets that are directly applicable, and which we directly apply, to the process of OOA. We have also learned a certain number of "spells" - ways to alter your perceived reality to help you achieve the specific goals that you want. Here, we use the term "spell" as a fun way to refer to mental programs that you can execute on Human Brain 1.0 (or compatible) to help you to achieve desirable results easily. Some of these spells we find we use quite consistently in the process of OOA, and we thought that it would be interesting and useful to you if we were to present a few of them here. Do treat the spells as a bit of fun - they are software for your mind! The spells will work whether you believe in them or not, but if you can approach them with curiosity and suspend *disbelief*, they will be even more effective. We also give follow-on references so that you can take your spell craft as far as you want it to go.

In summary, this book is very different to other books on OOA because (as far as we know) it is the first to explicitly present the *how*, the inner processes you go through in order to get the results you need. We set out to answer the question "how do you do OO analysis"? Your success as an OO analyst is the measure of how well we have managed to answer that question.

2. About the Website

The website for this book is:

www.secretsofanalysis.com

On the website is a collection of open source tools, created by us:

- Semantic Highlighter - a way of looking at text from different dimensions of meaning
- STAR modeler - a simple development environment for use cases, requirements and project glossaries

We created these tools for purely didactic reasons - to help us to get our ideas across to you in a more concrete way, and to allow you to easily apply the techniques that we present. They will also help you to do the exercises in the book.

Our inspiration for these tools is BlueJ - an integrated environment specifically designed for teaching Java. About a year ago, Jim gave a Java/XML course using BlueJ. He was blown away by the experience! He found that he could teach Java much more effectively using this

environment than he had ever been able to before. We wondered if it might be possible to create a simple tool set to teach some the ideas we present here. Our research led to the Semantic Highlighter and the STAR modeler. We hope you find them interesting and useful, and we'd love to get some feedback from you!

You will also find hyperlinks to the following open source tools (not written by us):

- FreeMind - a mind mapping tool
- CMap - a concept mapping tool

There are hyperlinks links to evaluation versions of the following tools:

- MagicDraw UML - our favourite UML modelling tool
- Visual Thesaurus - our favourite dictionary/thesaurus
- TreePad - our favourite text editor (this book was written using TreePad)

As well as tools, you will also find all of the artifacts for our worked example, the Orne Library Automated System (OLAS).

Enjoy!

3. Who should read this book

The key requirement for our readers is that you have some knowledge of UML and OO Analysis and Design (OOAD). This is not a book from which to learn UML or OOAD! Our other book, "UML 2 and the Unified Process" [Arlow 1] is the book for that.

Given the above prerequisite, readers may be:

1. Novice OO analyst/designers who need to learn how to do it
2. Experienced OO analyst/designers who are interested in our secrets
3. Programmers who have some UML and OO experience and who wish to learn how to become abstractionists
4. University students who are doing a course of OO analysis and design and who want to understand it more deeply

We should warn you that this book will change the way you think about things. If you don't want to change then you should not buy this book.

Our readers are creative and intelligent people who are interested in new ideas. This book is for them.

4. How to read this book

We usually like to write books that can be read in many different ways. For example there are at least five different ways to read "UML 2 and the Unified Process" [Arlow 1]. However, because this book is based around a worked example, the narrative is largely linear, and follows the development of the example. You should start at the beginning, and work your way through to the end. We develop our ideas as you progress through the book, so it's difficult, at first reading, to skip around too much. We have tried to pull out key ideas into notes, figures and a detailed summary at the end of each chapter, so you could just read these if you are very pushed for time. You will certainly get something out of this approach, but not as much as if you read the whole text.

We decided to base this book around a worked example because we felt that we needed to present our ideas about OOA in the context of real, but simple, OOA activities. This allows us

to make the presentation much more concrete, and we hope you will be able to see the immediate usefulness of the techniques we present. A different, technique-based structure for the book would have allowed you a more flexible approach to the text, but at the cost of losing the context.

5. Miskatonic University

The examples in this book are set in Miskatonic University by the side of the Miskatonic river in Arkham, Massachusetts. Miskatonic is a fictional University, by a fictional river in a fictional town invented by the (real) author H. P. Lovecraft.

Lovecraft (1890-1937) was an author of weird fiction who lived in Providence, Rhode Island, USA. Over the years he developed a fictional world in which to explore his ideas, that has been compared in richness with that of Tolkien's Lord of the Rings. However, rather than being set in a Middle Earth, most of Lovecraft's fiction is set in Massachusetts in fictional New England towns such as Arkham and Innsmouth. His stories are usually found classified under horror, but Lovecraft always viewed himself as a teller of weird tales rather than as a horror writer.

Jim has enjoyed reading Lovecraft over the years, and when we needed a setting for the example project in this book, Lovecraft's Massachusetts seemed like a rich and viable environment. We easily transferred our real experiences at Universities and businesses into this fertile fictional ground and this allowed us to tell of our experiences and lessons learned, without compromising client confidentiality. We also felt that the rather conservative and hide-bound Miskatonic University needed to be brought kicking and screaming (there's a lot of that already going on there) into the 21st century, with some state-of-the-art software systems.

Why choose a University for our setting? Firstly, a lot of our readers are likely to be students or to have been students in the not too distant past. The environment is therefore familiar to them. Secondly, a University provides the opportunity for many different systems that can be limited in scope to make the problems tractable in a book such as this. Finally, the system that we have chosen demonstrates features that are directly applicable to business systems. We feel no compelling need to present a book of models for business systems as we have already done that in "Enterprise Patterns and MDA" [Arlow 2]. The examples in this book are largely for instructional purposes.

Lovecraft inspired and encouraged a circle of writers such as Ramsay Campbell, August Derleth, Clark Ashton Smith and others to play in the fictional world that he created and to flesh out its structure. We are delighted to join them at play.

6. Neuro Linguistic Programming (NLP)

One of the cornerstones of this book is Neuro Linguistic Programming (NLP). This field was invented in the early 1970s by Dr. Richard Bandler, a mathematician, and Professor John Grinder, a linguist, at the University of California at Santa Cruz. They were working in collaboration with the social anthropologist, Gregory Bateson.

Neuro Linguistic Programming explores the how the mind (neuro) is influenced by, and influences, language (linguistic) patterns (programming). In [Bandler 1], Bandler and Grinder analyze patterns of therapeutic language and present that analysis as the NLP Meta Model – a model of human communication. In later work with the hypnotist Milton H. Erickson, Bandler presented the Milton Model – a model of hypnotic patterns of communication [Bandler 2].

Nowadays, you are more likely to be familiar with NLP as a "self-help" technique rather than as a communication science. To some extent, the field has been hijacked by self-help gurus.

This is a shame, because NLP has important things to say about communication, which is after all, the basis of every software engineering activity. From our particular perspective as analysts, NLP provides a rich set of communication strategies, the NLP Meta Model, that enables us get the accurate, high-quality information we need to do our jobs. This is primarily how we will be using NLP here. We have extended the NLP Meta Model of communication into M++ which is the Meta Model adapted specifically for OOA. This model gives you specific techniques for improving any communication. Clearly, this is an essential skill for every analyst.

Today, there are many different approaches to NLP. Jim learned NLP from Richard Bandler, and our approach is based on his. Being a hypnotist, Bandler regards trance as the underlying mechanism of NLP. And so do we. This is why trance, which is surely an unusual topic for an OOA book, is discussed here in some detail. By "trance", we simply mean your attention narrowing and focusing on a restricted set of (usually) internal events.

Prior to publication, one of our reviewers raised the interesting question "Is NLP scientific?" It's interesting because, although it sounds like a reasonable question, it is in fact meaningless noise until qualified (see Chapter 9). What particular theory of science should provide the evaluation criteria? And what specific aspects of NLP should be considered? If we assume that by "scientific", the reviewer means "backed up by an empirically verifiable theory", this strong definition excludes most of modern psychology, a lot of linguistics, many of the softer parts of computer science and even some physics (e.g. String Theory). For example, just think about the "science" of software requirements engineering or software project management. Is a complex application such as Microsoft Word formally verifiable, or is our faith (or lack thereof) in its operation really a matter of pragmatics and experience?

If we assume this definition of "scientific", NLP is such a broad field that the argument can go either way depending on which aspects of NLP you consider. As in many of the softer sciences, those research results that exist are often patchy, ambiguous and sometimes contradictory. Doubtless, much of the self-help content of NLP *is* "unscientific" even by the weakest definitions of the term. But it's always worth remembering that "unscientific" doesn't mean "doesn't work".

Going back to basics by cutting through the NLP self-help hype, we are left with a powerful set of techniques for effective communication. It is these linguistic patterns and techniques that we focus on here.

We are pragmatic software engineers and ultimately, if something works for us, we use it. We'll worry about *why* it works once we've got the system in on time and on budget.

The "spells" we present are definitely *not* scientific and that's why we call them spells. They are just a bit of fun that use some of the "self-help" aspects of NLP. Think of them as techniques and metaphors for manipulating your neurology for fun and profit. They may work for you, or they may not - you won't know until you try them out. We have clearly marked each spell so that those of our readers who prefer science to sensation may skip over them with no harm to either their world-view or to the rest of the text. The rest of you can enjoy them.