**MORGAN KAUFMANN - NEW BOOK PROPOSAL QUESTIONNAIRE**

Please fill out this form completely and email it directly to Todd Green at to.green@elsevier.com to have your proposal considered for a publishing agreement with Morgan Kaufmann.

**AUTHOR AND TITLE INFORMATION**

1. Tentative book title and subtitle (Try to include several alternate title/subtitle combinations. Make sure that your title is search-engine-optimized to enhance online discoverability of your book, but that it also accurately reflects the contents of the book.):

*Preferred title*

**Refactoring for Software Design Smells: A Pragmatic Approach to Manage Technical Debt**

*Alternative titles*

Refactoring for Software Design Smells: An Effective Practical Approach for Quality Design

Mastering Software Design Smells

Software Design Quality in Practice: Refactoring for Software Design Smells

1. Authors/Editors full name/s (As you would want them to appear on a book cover):

Girish Suryanarayana

Ganesh Samarthyam

Tushar Sharma

1. Position and affiliation (Job title and company/university):

Girish Suryanarayana – Senior Member Technical Staff, Siemens Corporate Research & Technologies, India

Ganesh Samarthyam – Independent Consultant and Trainer, Bangalore, India

Tushar Sharma – Lead Research Engineer, Siemens Corporate Research & Technologies, India

1. Author Biography (A single paragraph is best. Try to summarize your current work, areas of expertise, and relevant educational credentials as well as previous publications. Write this as you would if it were to appear in the front matter of your book and online in product descriptions):

**Girish Suryanarayana** is currently a senior member of technical staff at Siemens Corporate Research & Technologies, Bangalore, India. At Siemens, he is involved in providing architectural guidance to software development teams, pursuing research in topics related to software architecture and design, and conducting internal software design and architecture training. Girish is a member of the IEEE Software Advisory Board. He actively contributes to software engineering conferences. In 2013, he was on the program committee for Software Engineering In Practice (SEIP) track in International Conference on Software Engineering (ICSE). Girish received a PhD in information and computer science from the University of California, Irvine, in 2007. His research interests include software architecture, design patterns, design smells, refactoring, and reputation-based trust management systems. He is an IEEE-certified Software Engineering Certified Instructor (SECI) and regularly conducts training for the IEEE SWEBOK Certificate Program (SCP) and IEEE Certified Software Development Associate (CSDA) programs. He has also helped contribute course material for the IEEE’s SWEBOK Certificate Program (SCP). He is regularly invited by local universities to deliver guest lectures on software architecture and design topics. He can be reached at [girish.suryanarayana@gmail.com](mailto:girish.suryanarayana@gmail.com).

**Ganesh Samarthyam** has more than 11 years of experience in the IT industry. He is an independent consultant and corporate trainer based in Bangalore, India. Earlier, he worked for the Software Architecture and Development team in Siemens Corporate Research & Technologies in Bangalore. His work involved providing consultancy to development teams on code and design quality, pursuing applied research in the areas of software design, and conducting trainings on topics related to software design and refactoring and quality-driven development. Prior to Siemens, he worked in Hewlett-Packard's C++ compiler team for 5 years where he also represented HP in the ANSI/ISO C++ standardization committee. He is an IEEE-certified Software Engineering Certified Instructor (SECI) and regularly conducts training for the IEEE SWEBOK Certificate Program (SCP) and IEEE Certified Software Development Associate (CSDA) program. He has also helped contribute course material for the IEEE’s SWEBOK Certificate Program (SCP).  In his free time, he also regularly contributes to a column on software programming and design in “Open Source For You” magazine. His areas of interest include design smells and refactoring, bug patterns, software quality assessment methods, and programming languages. He can be reached at [sgganesh@gmail.com](mailto:sgganesh@gmail.com).

**Tushar Sharma** is a member of the Software Architecture and Development team in Siemens Corporate Research and Technologies India. His work at Siemens involves researching and providing consultation on topics related to software design, refactoring, testability, design patterns, and change impact analysis. He has also been involved in a number of internal design assessments which has afforded him the opportunity to observe at close design methodologies in practice. He has a Masters degree in Computer Science from the Indian Institute of Technology-Madras, India, where he specialized in design patterns and refactoring. Tushar has several patents related to software design and also maintains a blog on design and refactoring. He has recently co-authored the book “Oracle Certified Professional Java SE 7 Programmer Exams 1Z0-804 and 1Z0-805" published by Apress in 2013. He can be reached at [tusharsharma@ieee.org](mailto:ts.iitm@gmail.com).

1. Full mailing address/es (Where you would want royalty statements and other important documentation sent):

Girish Suryanarayana

2304, 21st Cross, K.R.Road,

Banashankari – II Stage

Bangalore – 560070

India

Ganesh Samarthyam

C-061, Classic Royale Garden,

Hennur Main Road, Hennur,

Bangalore – 560043

India

Tushar Sharma

2B-210, Shilpitha Splendour,

Mahadevapura, Outer Ring Road,

Chinnappa Layout,

Bangalore – 560 048

India

1. Authors/Editors phone, fax, email:

Name Telephone: Fax Email

|  |  |  |  |
| --- | --- | --- | --- |
| Girish Suryanarayana | +91 99458 41608 |  | [girish.suryanarayana@gmail.com](mailto:girish.suryanarayana@gmail.com) |
| Ganesh Samarthyam | +91 98801 64463 |  | [ganesh.samarthyam@gmail.com](mailto:ganesh.samarthyam@gmail.com) |
| Tushar Sharma | +91 97405 20454 |  | [tusharsharma@ieee.org](mailto:tusharsharma@ieee.org) |

1. If an edited book, approximately how many contributors do you expect to comprise the book?

NA

**SUBJECT MATTER**

1. Definition of topic (Please define the topic of the book in lay terms so that someone who has no context for your work can understand the subject matter):

In the medical domain, it is said, "A good doctor knows the medicines, but a great doctor is one who knows the disease". We believe a similar saying is applicable for software design - "A good designer is one who knows the various principles and patterns, but a great designer is one who knows the smells". It is only when a developer/designer is aware of design smells does he actually understand what he did wrong while designing, what design principles did he overlook or mis-apply, and what principles need to be applied properly to address those smells. In summary, it is very important for any software practitioner to be aware of design smells, violated design principle(s), and corresponding refactorings.

1. Brief description of topic which could be used for promotional purposes (Write one strong paragraph as it might appear on the back cover of the book and in online promotional copy.):

Capers Jones (a luminary of the software engineering community) conducted a study in 2012 across five large corporations and found that the number of *software defects that can be traced back to errors in software design* was *as high as 64%*! Hence, ensuring a high quality of software design is critical for developing high-quality software. However, as Fred Brooks (who received the Turing award in 1999) observes in his book “The Mythical Man Month”, software *design is an inherently complex activity*; the intangible nature of software makes it difficult for humans to envision, develop, and reason about its design. Towards this end, several software engineering design principles and practices have indeed been proposed to guide the development of high-quality software design. Yet, *most industrial software today still suffers from poor software design quality*. Our experience shows that this is primarily because these design principles are not well understood and therefore tend to be ignored in practice. For instance, developers and software engineers may "know" of principles and patterns, but are not aware of the "smells" that exist in their design because of wrong or mis-application of principles or patterns. These ***smells tend to contribute heavily to technical debt and need to be addressed via proper refactoring***. This is the cornerstone of our planned book. Using a number of examples, our book discusses ***25 structural design smells*, why they matter in the context of technical debt, and *potential refactoring solutions for these smells***. The detailed description of each smell and their corresponding refactoring allows them to be used as ***nuggets of readily usable knowledge***. If you are a software developer, designer, or an architect you will find high take away value when you read this book.

1. Outline your reasons for proposing a new book in this area (Why you? Why now? Why this book? What is your motivation?):

As part of our work, we regularly deal with large-scale and critical software. Our work in the area of software design can be categorized into three main activities.

1. As assessment experts – We have conducted numerous design and code assessments for projects differing in scope, size, kind, and domain. These experiences have given us practical insights on the underlying problematic aspects of software design.
2. As consultant architects – We are often called into a critical project to provide guidance on software architecture and design aspects. During such encounters, we have helped analyze a number of problems and issues arising from software design in real-world software products.
3. As trainers – We are part of a team that regularly conducts trainings on software architecture, design, and refactoring for novice as well as experienced software professionals.

Our involvement in these activities has given us the opportunity to interact with a large number of software designers and architects, observe them at work, see how they approach design, and finally realize the way smells creep into the design. Our experiences highlight a strong need for a comprehensive compilation of software design smells and the corresponding refactoring solutions for industry professionals.

However, when we thoroughly analyzed the existing books available in the market, we found that there is no single book that meets this need. For this reason, we started writing this book.

As part of the internal trainings and the assessments that we engage in at work, we have already been exposing software practitioners to the fundamental concepts of design smells and how they can be refactored. The feedback received from our training sessions and assessments has been very positive. Practitioners have reported that they have benefited from learning about smells and how to refactor them. Additionally, their project managers have also evinced a keen interest in this topic because of the potential reduction in eventual project cost! This feedback bolsters the value that our book can provide to practitioners.

Our initial work related to this book has been introduced in a paper titled [“*Towards a principle-based classification of structural design smells*”](http://www.jot.fm/issues/issue_2013_06/article1.pdf) which appeared in The Journal of Object Technology (JOT). Here is a quote from **Stéphane Ducasse** (associate editor of JOT) from his review of our paper: “Excellent work! This paper can be a key reference in the field.” He has also tentatively accepted our request to write a foreword for the book.

As support for our book, we also enclose below comments from two other reviewers who liked our work and suggested that it could be the basis of a book in the future.

**Reviewer 1** (when we submitted our earlier version of the paper to SPLASH) – “I do find this work interesting and I think it's the start of what could be an excellent book.”

**Reviewer 2** (when submitted to JOT) – “I would suggest you to edit a book of the full catalog :)”

1. List unique features of your book which will attract prospective buyers (3 – 5 concise bullets summarizing the things contained in the book that make it unique and worth reading):

* The book presents a **comprehensive catalogue** of structural design smells and is a single point of reference for structural design smells. The book discusses refactoring strategies for each design smell.
* The book illustrates **each smell with examples** **from OpenJDK** (Open Java Development Kit) **and commercial software** which makes the book an interesting read.
* Smells and their refactorings are **presented along with source code and visualized using UML diagrams** – this approach makes it easy to grasp the underlying concept.
* The book provides **interesting anecdotes** drawn from our experience about how certain smells are caused and difficulties in refactoring them. This not only makes it easy to absorb the concept but also provides practitioners with food-for-thought in the context of their projects. These anecdotes often reveal insights about other aspects of software engineering such as processes and people that have a significant impact on design quality and technical debt.

1. What are the benefits of this book for the reader (3 – 5 concise bullets summarizing what the reader will take away from reading the book)?
2. The book has a **comprehensive catalogue** for structural design smells and their refactoring solutions. This will help practitioners understand the kind of problems that could be occurring in their design and give them ideas on how they can refactor the smells.
3. This book guides the reader on **how to approach addressing technical debt** by identifying design smells and refactoring them.
4. This book shows **how to apply design principles to guide refactoring** process.
5. Please provide an outline including chapter titles, major subheadings within chapters, and a list of any supplementary material (such as a glossary) to appear at the end of the book. (You are encouraged to include a brief paragraph about the content of each chapter. If your proposed book is a revision please provide an annotated table of contents from the previous edition, so that reviewers can compare and evaluate the proposed changes between the two. If you do not posses your original table of contents, please contact me, and I can procure a copy).

Our objective is that the book should come across as a readily usable catalog of design smells and corresponding refactorings so that software developers can immediately start using it in their daily work. With this in mind, we plan to structure our book into 3 parts.

**Part I** (Chapters I – III) would cover background and introduction to technical debt, the role of refactoring in addressing technical debt, discussion on well-known design principles, introduction to structural design smells, and a principle-based classification for design smells.

**Part II** (Chapters IV – VII) would describe our catalog of 25 design smells that we have classified as violation of the four fundamental principles discussed by Grady Booch i.e. Abstraction, Encapsulation, Modularity, and Hierarchy. For each design smell, we will show examples from JDK where they occur, include anecdotes from real-world experience of the authors, describe why those smells are undesirable, discuss the possible refactoring techniques for that smell, and illustrate specific ways in which the JDK examples can be refactored to address that smell.

**Part III** (Chapter VIII) would discuss relationships between design smells including their trade-offs, their co-occurrence, and how they can combine to form a bigger problem.

Here is the proposed table of contents for the book:

|  |  |
| --- | --- |
| **1**  1.1  1.2  1.3  1.4  1.5  1.6 | **Technical Debt**  What is technical debt?  What constitutes technical debt?  Impact of technical debt  How does technical debt occur?  How to manage technical debt?  Role of refactoring in managing technical debt |
| **2**  2.1  2.2  2.3 | **Design Principles and Constraints**  Introduction  Design principles and impact on quality attributes  A list of design principles |
| **3**  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9 | **Design Smells**  Design smells  What are ‘design smells’?  Why care about smells?  What causes smells?  The scope of smells  Impact of smells on design quality  A classification of design smells  A naming scheme for smells  A template for documenting smells |
| **4**  4.1  4.2  4.3  4.4  4.5  4.6  4.7 | **Abstraction**  Incomplete Abstraction  Imperative Abstraction  Unutilized Abstraction  Missing Abstraction  Duplicate Abstraction  Multifaceted Abstraction  Unnecessary Abstraction |
| **5**  5.1  5.2  5.3 | **Encapsulation**  Lenient encapsulation  Missing encapsulation  Unexploited encapsulation |
| **6**  6.1  6.2  6.3  6.4  6.5  6.6 | **Modularization**  Broken Modularization  Unused Modularization  Insufficient Modularization  Split Modularization  Cyclic-Dependency Modularization  Hub-like Modularization |
| **7**  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8 | **Hierarchy**  Folded hierarchy  Missing hierarchy  Deep hierarchy  Rebellious hierarchy  Broken hierarchy  Cyclic hierarchy  Polygonal hierarchy  Unnecessary hierarchy |
| **8**  8.1  8.2  8.3  8.4 | **Interplay of Smells**  Compound smells  Co-existing smells  Competing smells  Finding and refactoring smells in industrial and open source projects |
| **Appendix A** | **Further Reading** |
| **Appendix B** | **List of Tools for Detecting Design Smells and Refactoring** |

# 

# **MANUSCRIPT INFORMATION**

1. How many illustrations, excluding tables, do you expect to have in the book?

Approx. 80 (we plan to have numerous – mostly hand-drawn – design diagrams to illustrate smells and their refactored solutions)

1. Approximately how many pages do you expect your book to be? (Most MK books average 500 words per page. Illustrations typically take up a half a page. Please take these factors into account when estimating.)

Approximately 350 pages.

1. List any special pedagogical features you expect to include (problems/solutions, exercises, tips, sidebars, etc.)

* The book illustrates **each smell with examples** **from OpenJDK** (Open Java Development Kit) **and commercial software** which makes the book an interesting read. Each smell description has a specific section devoted to examples of that smell. We make a conscious choice to not include refactoring suggestions immediately so that it gives readers a chance to ponder over the refactoring solution for those examples. The associated refactoring for those examples is only discussed at the end of the particular smell section. In a way, this layout of smell description resembles a problem/solution approach.
* Smells and their refactorings are **presented along with source code and visualized using UML diagrams** – this approach makes it easy to grasp the underlying concept.
* We include a few **cartoons relevant to the topic** (mostly from the popular [Geek & Poke](http://geek-and-poke.com/) cartoons) to make the book an interesting read.
* The book provides **interesting anecdotes** drawn from our experience about how certain smells are caused and difficulties in refactoring them. This not only makes it easy to absorb the concept but also provides practitioners with food-for-thought in the context of their projects. These anecdotes also provide practical tips on aspects related to managing processes and people better in order to address technical debt. These anecdotes will be presented as sidebars in the book.

1. What software do you intend to use to prepare your manuscript? (Note that MS Word is the preferred file format for manuscript preparation. LaTex is also acceptable but may take longer to typeset and process, increasing time to market.)

MS Word (2007 version)

1. Please describe any images, slide decks, sample course syllabi, source code, software, or other materials for a companion web page that will accompany the text. (MK will host these ancillaries on our web site. You should plan to deliver all ancillaries when you deliver the final manuscript.)

We will create a slide-deck on the content of the book. We plan to use this for “Refactoring for Design Smells” tutorials and trainings that we may do in the future. These can be shared on the accompanying web page.

We are also currently maintaining a blog on aspects related to software design, refactoring, and smells. Content from the articles on the blog can be used on the accompanying web page.

1. When do you intend to start writing and when do you plan to complete your final manuscript (Please be as realistic as possible.)?

We plan to complete the **first draft of the book by November 30th 2013.**

We plan to deliver the **final manuscript by December 31st 2013**.

**THE MARKET**

1. Identify the readership for your book
   1. PRIMARY MARKET (Include level—undergraduate, graduate, professional, reference—as well as job titles of prospective readers. If you have data available on the market size for job titles or university courses, please include that information. Remember that not everyone is your target audience. Don’t include people who you think *should* read your book, but rather include who *will* buy your book.):

* Software professionals including **software engineers**, **technical leads**, **software designers**, and **software** **architects**, and managers working on any kind of software
* Software professionals working on developing design and code analyzer tools
* Corporate trainers for topics related to software design, refactoring, design smells, and design quality
  1. SECONDARY MARKET (include level—undergraduate, graduate, professional, reference—as well as job titles of prospective readers. You can be a bit broader and more general here. Include everyone who you think would be interested in reading your book.):
* Researchers/academicians working in the area of software design, refactoring, or change impact analysis
* Students who may find this text useful as a supplementary text for object oriented design course

This book could be used as a supplementary text for graduate or professional level courses on software architecture, software design, refactoring, and object oriented programming.

* 1. INTERNATIONAL or GLOBAL MARKET (Are there any geographical limitations to the value of the content in your book? Are there any markets that might have a particularly high demand for this information?):

1. Indicate any societies, professional organization, companies or other groups that might purchase your book in quantity (provide hyperlinks and/or names/emails of your contacts at each.):

One of the authors Dr. Girish Suryanarayana runs a high-profile software design and architecture training for Siemens employees. Design Smells is an integral part of this training, and the expectation is that if there is a book on Design Smells, it will be purchased by Siemens in bulk for these training sessions.

Like Siemens, other large software organizations such as Infosys, Hewlett-Packard, TCS, and Wipro Infotech have dedicated training divisions that conduct regular trainings for their employees on topics such as software design, software quality, and refactoring. Since our book offers practical and hands-on guidance on improving software design quality, it would be relevant for these training divisions to procure a large number of copies of our book.

1. List relevant journals and their publishers (provide hyperlinks):

**Widely-read magazines:** [IEEE Software](http://www.computer.org/software‎), [Communications of the ACM](cacm.acm.org/) (CACM), and [Dr Dobb’s Journal](http://www.drdobbs.com/‎)

1. List key meetings/conferences in this area (provide hyperlinks and dates/locations for the next conference):

* SIGPLAN Conferences on Object-Oriented Programming, Systems, Languages, and Applications ([SPLASH](http://splashcon.org/2014)); organized in Portland US in 2014
* European Conference on Object-Oriented Programming ([ECOOP](ecoop14.it.uu.se/)); organized in Uppsala, Sweden in 2014
* Pattern Languages of Programs ([PLoP](http://www.hillside.net/plop/))
* International Conference for Software Engineering ([ICSE](2014.icse-conferences.org/)); organized in Hyderabad, India in 2014
* Foundations of Software Engineering ([FSE](fse22.gatech.edu/)); organized in Hong Kong, China in 2014
* International Conference of Software Maintenance ([ICSM](http://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=31980)); organized in Victoria BC, Canada in 2014

1. Are there training needs that the book may satisfy? If so, identify the industry, representative companies, and the type of training involved. Does your book satisfy a college course need? If so, list the course title(s), department(s), and applications, along with estimated enrollments. Indicate whether the course is usually required and, if not, how many schools offer it.

Large software organizations such as Infosys, Hewlett-Packard, TCS, and Siemens have dedicated training divisions that conduct regular trainings for their employees on topics such as software design, software quality, and refactoring. Since our book offers practical guidance on improving design quality and avoiding technical debt by focusing on refactoring for design smells, our book may satisfy their needs.

**COMPETITION**

1. Please list in order of importance any books that compete directly with or are similar to your book. (If you are inclined to answer “none” to this question, please tell us where your intended audience currently gets information about this topic…e.g., conferences, tutorials, journal articles, web forums, etc…please be as specific as you can. Try to include the most related book titles that are available, even if none directly compete. If one has a chapter on the topic of your book, for example, please include it! It is best to be exhaustive here! Add more rows if necessary. In the notes column, please discuss what makes your book different and why someone would purchase yours instead.):
2. Provide an outline of the main ways your book is better than and differs from the books mentioned above:

|  |  |
| --- | --- |
| **Title** | AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis |
| **Author(s)** | William J. Brown, Raphael C. Malveau, Hays W. McCormick, Thomas J. Mowbray |
| **ISBN** | 978-0471197133 |
| **Publisher** | Wiley |
| **Publication Date** | April 3, 1998 |
| **Page Count** | 336 |
| **List Price** | $60 |
| **Comments** | This book discusses anti-patterns in software projects, software architectures, and in software project management. Amazon sales rank (as on 23rd October) is #236,243.  Our proposed book and this book are similar from the viewpoint of discussing ineffective design solutions and how they could be addressed. However, our proposed book is focused on structural design smells (i.e., design-level structural anti-patterns), which the competing book does not address. Hence, this book is not a direct competition to our proposed book, but substantiates the argument that a book on design smells and refactoring is relevant and important to readers. |

|  |  |
| --- | --- |
| **Title** | Refactoring: Improving the Design of Existing Code |
| **Author(s)/**  **Editor(s)** | Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts |
| **ISBN** | 978-0201485677 |
| **Publisher** | Addison-Wesley Professional |
| **Publication Date** | July 8, 1999 |
| **Page Count** | 464 |
| **List Price** | $44 |
| **Comments** | This book is a best-seller on code smells and refactoring. The Amazon sales rank for the book (as on 23rd October) is #15,024.  With this book, the terms “refactoring” and “bad smells” has become part of developer’s vocabulary. The term “refactoring” refers to structural improvement of code and the term “bad smells” refer to indicators of problems in design. The book presents a set of “code smells” (though the term “code” is used in this term, it refers to code-level and some design-level smells) and discusses refactoring solutions to improve structural design.  Our proposed book and this competing book are similar in that both the books focus on smells. However, the focus of our book is on “design-level smells” whereas the focus of the competing book is on refactoring with additional discussion on “code-level and a few design-level smells”. Given the importance of understanding “bad smells”, we feel that Fowler’s book missed out an opportunity to cover design smells in a more comprehensive manner. In our book, we provide an in-depth discussion on design smells and **complement** Fowler’s work on refactoring. |

|  |  |
| --- | --- |
| **Title** | Object Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems |
| **Author(s)/**  **Editor(s)** | Michele Lanza, Radu Marinescu, Stephane Ducasse |
| **ISBN** | 978-3540244295 |
| **Publisher** | Springer |
| **Publication Date** | September 14, 2006 |
| **Page Count** | 220 |
| **List Price** | $90 |
| **Comments** | This excellent book discusses using object oriented metrics for detecting “design flaws”. The Amazon sales rank for the book (as on 23rd October) is #3,057,127. The main focus of the book is to describe “detection strategies” for detecting design flaws using various OO metrics. The book also shows an approach for visualizing design flaws and also discusses strategies to refactor the flaws.  Our book and this competing book are similar in that the focus is on flaws or smells in design. However, unlike the competing book, the main focus of our book is on **refactoring** **for smells**. A minor drawback of this competing book is that it misses out some key smells, for instance, encapsulation related smells. The major drawback of the competing book is that it is written in an academic language and follows an approach which is perhaps not very appealing to practitioners. In contrast, our book has numerous real-world examples, has anecdotes from actual experiences, and is written in a way that is more appealing to software professionals. The practice-oriented approach taken by Fowler in his “refactoring” book shows that it is feasible to write a bestseller on this topic in a way that is accessible and appealing to practitioners. |

|  |  |
| --- | --- |
| **Title** | Design Patterns: Elements of Reusable Object-Oriented Software |
| **Author(s)/**  **Editor(s)** | Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides |
| **ISBN** | 978-0201633610 |
| **Publisher** | Addison-Wesley Professional |
| **Publication Date** | November 10, 1994 |
| **Page Count** | 416 |
| **List Price** | $60 |
| **Comments** | This is a landmark book which bought “design patterns” to mainstream in software engineering. The Amazon sales rank for the book (as on 23rd October) is #4,194.  This book presents a catalogue of 23 design patterns and classifies them into creational, structural, and behavioural patterns. The main contribution of this book is the identification of design patterns, cataloguing them, and presenting detailed discussion on each of these patterns.  Our proposed book and this classic book are similar in that both books aim to condense design knowledge in the form of readily usable catalogues. The focus of the competing book is on introducing a set of design patterns whereas the focus of our book is on **explicitly identifying common ineffective solutions** and how they can be refactored. Refactoring many of the design smells would require introducing design patterns. Due to the large number of design smells documented in literature, our book limits its focus to discussing only structural design smells and their refactoring solutions (and does not discuss other design smells such as behavioural smells). |

|  |  |
| --- | --- |
| **Title** | Refactoring in Large Software Projects: Performing Complex Restructurings Successfully |
| **Author(s)/**  **Editor(s)** | Martin Lippert, Stephen Roock |
| **ISBN** | 978-0470858929 |
| **Publisher** | Wiley |
| **Publication Date** | May 18, 2006 |
| **Page Count** | 286 |
| **List Price** | $60 |
| **Comments** | The Amazon sales rank for the book (as on 23rd October) is #2,066,562. This book focuses on architectural smells and refactoring. Since this book is written by consultants, it is a practical book focusing on addressing real-world problems in performing refactoring at architectural level. Since large-scale refactoring is still an emerging topic and this topic is still not well-known in the software engineering community, this book may not be selling well.  Our proposed book differs from this competing book in many ways. The focus of our book is on **design** **smells** and not architectural smells. Our book views refactoring of smells in the context of “technical debt”, which is currently receiving considerable focus by the entire SE community. The competing book focuses on a niche audience – architects. However, our book is on detailed design, which is of interest to anyone involved in programming and design - experienced programmers, leads, and architects. The wider scope of our book translates to a wider (but still focused) target audience, which we believe will positively impact the sales of the book. |

|  |  |
| --- | --- |
| **Title** | Managing Software Debt: Building for Inevitable Change |
| **Author(s)/**  **Editor(s)** | Chris Sterling |
| **ISBN** | 978-0321554130 |
| **Publisher** | Addison-Wesley Professional |
| **Publication Date** | December 20, 2010 |
| **Page Count** | 288 |
| **List Price** | $45.99 |
| **Comments** | The Amazon sales rank for the book (as on 23rd October) is #324,142. This book looks at “technical debt” in the context of Agile methods. It discusses various kinds of debt, why they occur, how they can be addressed, etc using Agile approach.  The competing book is one of the few practical books on managing technical debt. However, this book will interest those managing Agile projects rather than engineers or designers in software development teams. For this reason, the competing book has a niche target audience. Compared to this competing book, our proposed book views technical debt from a technical perspective. Our book focuses on important design smells and corresponding refactoring – and hence the take-away value of our book for software engineers is high. Our book is of interest to engineers of various levels of experience in software development teams and hence has wider target audience. |

|  |  |
| --- | --- |
| **Title** | Refactoring to Patterns |
| **Author(s)** | Joshua Kerievsky |
| **ISBN** | 978-0321213358 |
| **Publisher** | Addison-Wesley Professional |
| **Publication Date** | August 15, 2004 |
| **Page Count** | 400 |
| **List Price** | 64.99 |
| **Comments** | Amazon sales rank (as on 23rd October) is #234,117. This excellent and practical book on refactoring supplements Fowler’s classic book on Refactoring – this book shows how many of the code smells can be refactored to patterns. This book uses an example driven approach – the author takes real-world code with smells and provides step-by-step description of how the code can be refactored.  Our book is similar to this competing book in many ways - both focus on refactoring smells and use an example-driven approach. In fact, often the correct refactoring for our design smell is the use of an appropriate design pattern, a fact that is highlighted several times in our book. The difference lies in the **scope** – our book discusses ***design*** smells whereas Kerievsky’s discussion is centred mostly on ***code*** smells. Another difference is that we use examples from a real-world API i.e. OpenJDK extensively for illustration whereas this competing book uses general examples with only a few examples from JDK. Further, not all smells can be refactored to patterns and hence our book is more general than Kerievsky’s book. |

**REVIEWS**

1. Before submitting your proposal for internal approval, we must have the proposal reviewed by subject matter experts. Please list several people (including affiliations and email addresses if available) whose comments on your proposal you would value. These comments are important for our internal evaluation of your proposal. The more people you can suggest, the better, since not everybody we approach will take on the task and, sadly, not everyone who agrees to take on the task will complete the effort in a timely manner. Ideally, we like to use a mix of reviewers from both academia and industry, assuming your book has an audience in both communities; also, we like to use a geographically diverse (globally) group of reviewers as well. Please provide at least 5 – 10 possible reviewers and comment on why you would value their feedback. List them in preference order and include a line of notes describing why you think their feedback would be valuable.

* **Diomidis D. Spinellis** – Professor in the Athens University of Economics and Business; Author of Code Quality book; Creator of tools such as CKJM for OO metrics – [dds@aueb.gr](mailto:dds@aueb.gr)
* **Peter Sommerlad** – Director IFS Institute for Software; Well-known for his work on refactoring tools and patterns – [peter.sommerlad@hsr.ch](mailto:peter.sommerlad@hsr.ch)
* **Stephane Ducasse** – Research director at INRIA Lille; Researcher in the areas of software maintenance & evolution and language design; Author of many books including Object-Oriented Reengineering Patterns – [StephaneDucasse@inria.fr](mailto:StephaneDucasse@inria.fr)
* **Danny Dig** – Assistant Professor in Oregon State University; Significant contributions to refactoring community – [digd@eecs.oregonstate.edu](mailto:digd@eecs.oregonstate.edu)
* **Linda Rising** – CIO of the Hillside Group; Author, lecturer, independent consultant; well-known for her pattern’s related work – [risingl@tds.net](mailto:risingl@tds.net)
* **Michele Lanza** – Professor in University of Lugano; Co-author of Object Oriented Metrics in Practice book – [michele.lanza@usi.ch](mailto:michele.lanza@usi.ch)
* **Oscar M. Nierstrasz** – Professor in University of Bern; Author of many books including Object-Oriented Reengineering Patterns book – [oscar@iam.unibe.ch](mailto:oscar@iam.unibe.ch)
* **William C. Wake** – Senior consultant with Industrial Logic; Author of the book Refactoring Workbook – [William.Wake@acm.org](mailto:William.Wake@acm.org)
* **Joshua Kerievsky** – CEO of Industrial Logic; Author of Refactoring to Patterns book – [joshua@industriallogic.com](mailto:joshua@industriallogic.com)

Researchers working in the area of Refactoring are listed in this webpage <http://refactoring.info/people.html>. We believe that these researchers will be good reviewers of our book as well.