1.	Record Nr.	UNINA9910453261903321
	Autore	Makkai Adam
	Titolo	Idiom structure in English / / by Adam Makkai
	Pubbl/distr/stampa	The Hague : , : Mouton, , 1972
	ISBN	3-11-081267-3
	Edizione	[Reprint 2013]
	Descrizione fisica	1 online resource (380 p.)
	Collana	Janua Linguarum. Series Maior ; ; 48 Janua linguarum. Series maior ; ; 48
	Disciplina	425
	Soggetti	English language - Idioms Stratificational grammar Electronic books.
	Lingua di pubblicazione	Inglese
	Formato	Materiale a stampa
	Livello bibliografico	Monografia
	Note generali	Revised version of the author's 1965 Yale University doctoral dissertation.
	Nota di bibliografia	Includes bibliographical references.
	Nota di contenuto	Frontmatter -- PREFACE -- TABLE OF CONTENTS -- ABBREVIATIONS -- 0. INTRODUCTION : SCOPE OF THE PRESENT STUDY -- PART ONE -- 1. THEORETICAL CONSIDERATIONS -- PART TWO -- II. A PARTIAL CLASSIFICATION OF SOME OF THE MOST FREQUENT TYPES OF LEXEMIC IDIOMS IN STANDARD AMERICAN ENGLISH -- APPENDIX -- BIBLIOGRAPHY -- AUTHOR INDEX -- TOPICAL INDEX

2.	Record Nr.	UNINA9910788070503321
	Autore	Suryanarayana Girish
	Titolo	Refactoring for software design smells : managing technical debt / / Girish Suryanarayana, Ganesh Samarthyam, Tushar Sharma
	Pubbl/distr/stampa	Waltham, Massachusetts ; : , : Morgan Kaufmann, , 2015 ©2015
	ISBN	0-12-801646-9
	Edizione	[1st edition]
	Descrizione fisica	1 online resource (259 p.)
	Disciplina	005.1/6
	Soggetti	Software refactoring Software failures
	Lingua di pubblicazione	Inglese
	Formato	Materiale a stampa
	Livello bibliografico	Monografia
	Note generali	Description based upon print version of record.
	Nota di bibliografia	Includes bibliographical references and index.
	Nota di contenuto	FrontCover; Refactoring forSoftware DesignSmells; Copyright; Dedication; Contents; Foreword by Grady Booch; Foreword by Dr. Stephane Ducasse; Preface; WHAT IS THIS BOOK ABOUT?; WHAT DOES THIS BOOK COVER?; WHO SHOULD READ THIS BOOK?; WHAT ARE THE PREREQUISITES FOR READING THIS BOOK?; HOW TO READ THIS BOOK?; WHERE CAN I FIND MORE INFORMATION?; WHY DID WE WRITE THIS BOOK?; Acknowledgments; Chapter 1 - Technical Debt; 1.1 WHAT IS TECHNICAL DEBT?; 1.2 WHAT CONSTITUTES TECHNICAL DEBT?; 1.3 WHAT IS THE IMPACT OF TECHNICAL DEBT?; 1.4 WHAT CAUSES TECHNICAL DEBT?; 1.5 HOW TO MANAGE TECHNICAL DEBT? Chapter 2 - Design Smells2.1 WHY CARE ABOUT SMELLS?; 2.2 WHAT CAUSES SMELLS?; 2.3 HOW TO ADDRESS SMELLS?; 2.4 WHAT SMELLS ARE COVERED IN THIS BOOK?; 2.5 A CLASSIFICATION OF DESIGN SMELLS; Chapter 3 - Abstraction Smells; 3.1 MISSING ABSTRACTION; 3.2 IMPERATIVE ABSTRACTION; 3.3 INCOMPLETE ABSTRACTION; 3.4 MULTIFACETED ABSTRACTION; 3.5 UNNECESSARY ABSTRACTION; 3.6 UNUTILIZED ABSTRACTION; 3.7 DUPLICATE ABSTRACTION; Chapter 4 - Encapsulation Smells; 4.1 DEFICIENT ENCAPSULATION; 4.2 LEAKY ENCAPSULATION; 4.3 MISSING ENCAPSULATION; 4.4 UNEXPLOITED ENCAPSULATION; Chapter 5 - Modularization Smells 5.1 BROKEN MODULARIZATION5.2 INSUFFICIENT MODULARIZATION; 5.3 CYCLICALLY-DEPENDENT MODULARIZATION; 5.4 HUB-LIKE

MODULARIZATION; Chapter 6 - Hierarchy Smells; 6.1 MISSING HIERARCHY; 6.2 UNNECESSARY HIERARCHY; 6.3 UNFACTORED HIERARCHY; 6.4 WIDE HIERARCHY; 6.5 SPECULATIVE HIERARCHY; 6.6 DEEP HIERARCHY; 6.7 REBELLIOUS HIERARCHY; 6.8 BROKEN HIERARCHY; 6.9 MULTIPATH HIERARCHY; 6.10 CYCLIC HIERARCHY; Chapter 7 - The Smell Ecosystem; 7.1 THE ROLE OF CONTEXT; 7.2 INTERPLAY OF SMELLS; Chapter 8 - Repaying Technical Debt in Practice; 8.1 THE TOOLS; 8.2 THE PROCESS; 8.3 THE PEOPLE
Appendix A - Software Design PrinciplesA.1 ABSTRACTION; A.2 ACYCLIC DEPENDENCIES PRINCIPLE; A.3 DON'T REPEAT YOURSELF PRINCIPLE; A.4 ENCAPSULATION; A.5 INFORMATION HIDING PRINCIPLE; A.6 KEEP IT SIMPLE SILLY; A.7 LISKOV'S SUBSTITUTION PRINCIPLE; A.8 HIERARCHY; A.9 MODULARIZATION; A.10 OPEN/CLOSE PRINCIPLE; A.11 SINGLE RESPONSIBILITY PRINCIPLE; A.12 VARIATION ENCAPSULATION PRINCIPLE; Appendix B - Tools for Repaying Technical Debt; Appendix C - Notations for Figures; Appendix D - Suggested Reading; D.1 ESSENTIALS; D.2 REFACTORING AND REENGINEERING; D.3 PATTERNS AND ANTI-PATTERNS
D.4 TECHNICAL DEBTBibliography; Index

Sommario/riassunto

Awareness of design smells - indicators of common design problems - helps developers or software engineers understand mistakes made while designing, what design principles were overlooked or misapplied, and what principles need to be applied properly to address those smells through refactoring. Developers and software engineers may ""know"" 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 - further time owed to fix projects thought to b