Read Free Free Domain Specific Languages By Martin Fowler

Getting the books **Free Domain Specific Languages By Martin Fowler** now is not type of challenging means. You could not solitary going in the same way as ebook amassing or library or borrowing from your associates to edit them. This is an utterly easy means to specifically get guide by on-line. This online declaration Free Domain Specific Languages By Martin Fowler can be one of the options to accompany you taking into account having new time.

It will not waste your time. receive me, the e-book will definitely atmosphere you additional thing to read. Just invest tiny get older to entry this on-line proclamation **Free Domain Specific Languages By Martin Fowler** as with ease as review them wherever you are now.

WISE POWELL

DSL Engineering Simon and Schuster
This book covers several topics related to domain-specific language (DSL) engineering in general and how they can be handled by means of the JetBrains Meta Programming System (MPS), an open source language workbench developed by JetBrains over the last 15 years. The book begins with an overview of the domain of language workbenches, which provides perspectives and motivations underpinning the creation of MPS. Moreover, technical details of the language underneath MPS together with the definition of the tool's main features are discussed. The remaining ten chapters are then organized in three parts, each dedicated to a specific aspect of the topic. Part I "MPS in Industrial Applications" deals with the challenges and inadequacies of general-purpose languages used in companies, as opposed to the reasons why DSLs are essential, together with their benefits and efficiency, and summarizes lessons learnt by using MPS. Part II about "MPS in Research Projects" covers the benefits of text-based languages, the design and development of gamification applications, and research fields with generally low expertise in language engineering. Eventually, Part III focuses on "Teaching and Learning with MPS" by discussing the organization of both commercial and academic courses on MPS. MPS is used to implement languages for real-world use. Its distinguishing feature is projectional editing, which supports practically unlimited language extension and composition possibilities as well as a flexible mix of a wide range of textual, tabular, mathematical and graphical notations. The number and diversity of the presented use-cases demonstrate the strength and malleability of the DSLs defined using MPS. The selected contributions represent the current state of the art and practice in using JetBrains MPS to implement languages for real-world applications.
Practical Aspects of Declarative Languages Addison-Wesley Professional
Learn how to implement a DSL with Xtext and Xtend using easy-to-understand examples and best practices About This Book Leverage the latest features of Xtext and Xtend to develop a domain-specific language. Integrate Xtext with popular third party IDEs and get the best out of both worlds. Discover how to test a DSL implementation and how to customize runtime and IDE aspects of the DSL Who This Book Is For This book is targeted at programmers and developers who want to create a domain-specific language with Xtext. They should have a basic familiarity with Eclipse and its functionality. Previous experience with compiler implementation can be helpful but is not necessary since this book will explain all the development stages of a DSL. What You Will Learn Write Xtext grammar for a DSL; Use Xtend as an alternative to Java to write cleaner, easier-to-read, and more maintainable code; Build your Xtext DSLs easily with Maven/Tycho and Gradle; Write a code generator and an interpreter for a DSL; Explore the Xtext scoping mechanism for symbol resolution; Test most aspects of the DSL implementation with JUnit; Understand best practices in DSL implementations with Xtext and Xtend; Develop your Xtext DSLs using Continuous Integration mechanisms; Use an Xtext editor in a web application In Detail Xtext is an open source Eclipse framework for implementing domain-specific languages together with IDE functionalities. It lets you implement languages really quickly; most of all, it covers all aspects of a complete language infrastructure, including the parser, code generator, interpreter, and more. This book will enable you to implement Domain Specific Languages (DSL) efficiently, together with their IDE tooling, with Xtext and Xtend. Opening with brief coverage of Xtext features involved in DSL implementation, including integration in an IDE, the book will then introduce you to Xtend as this language will be used in all the examples throughout the book. You will then explore the typical programming development workflow with Xtext when we modify the grammar of the DSL. Further, the Xtend programming language (a fully-featured Java-like language tightly integrated with Java) will be introduced. We then explain the main concepts of Xtext, such as validation, code generation, and customizations of runtime and UI aspects. You will have learned how to test a DSL implemented in Xtext with JUnit and will progress to advanced concepts such as type checking and scoping. You will then integrate the typical Continuous Integration systems built in to Xtext DSLs and familiarize yourself with Xbase. By the end of the book, you will manually maintain the EMF model for an Xtext DSL and will see how an Xtext DSL can also be used in IntelliJ. Style and approach A step-by step-tutorial with illustrative examples that will let you master using Xtext and implementing DSLs with its custom language, Xtend.
Dependency Injection Principles, Practices, and Patterns Springer Nature
Domain-Specific LanguagesPearson Education
Proceedings of the 34th Annual Hawaii International Conference on System Sciences Springer Science & Business Media
Domain-Specific Languages (DSLs)--languages geared to specific vertical or horizontal areas of interest--are generating growing excitement from software engineers and architects. DSLs bring new agility to the creation and evolution of software, allowing selected design aspects to be expressed in terms much closer to the system requirements than standard program code, significantly reducing development costs in large-scale projects and product lines. In this breakthrough book, four leading experts reveal exactly how DSLs work, and how you can make the most of them in your environment. With Domain-Specific Development with Visual Studio DSL Tools , you'll begin by mastering DSL concepts and techniques that apply to all platforms. Next, you'll discover how to create and use DSLs with the powerful new

Microsoft DSL Tools--a toolset designed by this book's authors. Learn how the DSL Tools integrate into Visual Studio--and how to define DSLs and generate Visual Designers using Visual Studio's built-in modeling technology. In-depth coverage includes Determining whether DSLs will work for you Comparing DSLs with other approaches to model-driven development Defining, tuning, and evolving DSLs: models, presentation, creation, updates, serialization, constraints, validation, and more Creating Visual Designers for new DSLs with little or no coding Multiplying productivity by generating application code from your models with easy-to-use text templates Automatically generating configuration files, resources, and other artifacts Deploying Visual Designers across the organization, quickly and easily Customizing Visual Designers for specialized process needs List of Figures List of Tables Foreword Preface About the Authors Chapter 1 Domain-Specific Development Chapter 2 Creating and Using DSLs Chapter 3 Domain Model Definition Chapter 4 Presentation Chapter 5 Creation, Deletion, and Update Behavior Chapter 6 Serialization Chapter 7 Constraints and Validation Chapter 8 Generating Artifacts Chapter 9 Deploying a DSL Chapter 10 Advanced DSL Customization Chapter 11 Designing a DSL Index

Hands-On Enterprise Automation with Python Pragmatic Bookshelf

This tutorial book presents an augmented selection of material presented at the International Summer School on Generative and Transformational Techniques in Software Engineering, GTTSE 2005. The book comprises 7 tutorial lectures presented together with 8 technology presentations and 6 contributions to the participants workshop. The tutorials combine foundations, methods, examples, and tool support. Subjects covered include feature-oriented programming and the AHEAD tool suite; program transformation with reflection and aspect-oriented programming, and more.

Generative and Transformational Techniques in Software Engineering Simon and Schuster

"[The authors] are pioneers. . . . Few in our industry have their breadth of knowledge and experience." —From the Foreword by Dave Thomas, Bedarra Labs Domain-Specific Modeling (DSM) is the latest approach to software development, promising to greatly increase the speed and ease of software creation. Early adopters of DSM have been enjoying productivity increases of 500–1000% in production for over a decade. This book introduces DSM and offers examples from various fields to illustrate to experienced developers how DSM can improve software development in their teams. Two authorities in the field explain what DSM is, why it works, and how to successfully create and use a DSM solution to improve productivity and quality. Divided into four parts, the book covers: background and motivation; fundamentals; in-depth examples; and creating DSM solutions. There is an emphasis throughout the book on practical guidelines for implementing DSM, including how to identify the necessary language constructs, how to generate full code from models, and how to provide tool support for a new DSM language. The example cases described in the book are available the book's Website, www.dsmbook.com, along with, an evaluation copy of the MetaEdit+ tool (for Windows, Mac OS X, and Linux), which allows readers to examine and try out the modeling languages and code generators. Domain-Specific Modeling is an essential reference for lead developers, software engineers, architects, methodologists, and technical managers who want to learn how to create a DSM solution and successfully put it into practice.

Variable Domain-specific Software Languages with DjDSL Springer

Written by the creator of the Unicon programming language, this book will show you how to implement programming languages to reduce the time and cost of creating applications for new or specialized areas of computing Key FeaturesReduce development time and solve pain points in your application domain by building a custom programming languageLearn how to create parsers, code generators, file readers, analyzers, and interpretersCreate an alternative to frameworks and libraries to solve domain-specific problemsBook Description The need for different types of computer languages is growing rapidly and developers prefer creating domain-specific languages for solving specific application domain problems. Building your own programming language has its advantages. It can be your antidote to the ever-increasing size and complexity of software. In this book, you'll start with implementing the frontend of a compiler for your language, including a lexical analyzer and parser. The book covers a series of traversals of syntax trees, culminating with code generation for a bytecode virtual machine. Moving ahead, you'll learn how domain-specific language features are often best represented by operators and functions that are built into the language, rather than library functions. We'll conclude with how to implement garbage collection, including reference counting and mark-and-sweep garbage collection. Throughout the book, Dr. Jeffery weaves in his experience of building the Unicon programming language to give better context to the concepts where relevant examples are provided in both Unicon and Java so that you can follow the code of your choice of either a very high-level language with advanced features, or a mainstream language. By the end of this book, you'll be able to build and deploy your own domain-specific languages, capable of compiling and running programs. What you will learnPerform requirements analysis for the new language and design language syntax and semanticsWrite lexical and context-free grammar rules for common expressions and control structuresDevelop a scanner that reads source code and generate a parser that checks syntaxBuild key data structures in a compiler and use your compiler to build a syntax-coloring code editorImplement a bytecode interpreter and run bytecode generated by your compilerWrite tree traversals that insert information into the syntax treeImplement garbage collection in your languageWho this book is for This book is for software developers interested in the idea of inventing their own language or developing a domain-specific language. Computer science students taking compiler construction courses will also find this book highly useful as a practical guide to language implementation to supplement more theoretical textbooks. Intermediate-level knowledge and experience working with a high-level language such as Java or the C++ language are expected to help you get the most out of this book.

DSLs in Boo Springer Nature

Summary Dependency Injection Principles, Practices, and Patterns teaches you to use DI to reduce hard-coded dependencies between application components. You'll start by learning what DI is and what types of applications will benefit from it. Then, you'll work through concrete scenarios using C# and the .NET framework to implement DI in your own projects. As you dive into the thoroughly-explained examples, you'll develop a foundation you can apply to any of the many DI libraries for .NET and .NET Core. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Dependency Injection (DI) is a great way to reduce tight coupling between software components. Instead of hard-coding dependencies, such as specifying a database driver, you make those connections through a third party. Central to application frameworks like ASP.NET Core, DI enables you to better manage changes and other complexity in your software. About the Book Dependency Injection Principles, Practices, and

Patterns is a revised and expanded edition of the bestselling classic Dependency Injection in .NET. It teaches you DI from the ground up, featuring relevant examples, patterns, and anti-patterns for creating loosely coupled, well-structured applications. The well-annotated code and diagrams use C# examples to illustrate principles that work flawlessly with modern object-oriented languages and DI libraries. What's Inside Refactoring existing code into loosely coupled code DI techniques that work with statically typed OO languages Integration with common .NET frameworks Updated examples illustrating DI in .NET Core About the Reader For intermediate OO developers. About the Authors Mark Seemann is a programmer, software architect, and speaker who has been working with software since 1995, including six years with Microsoft. Steven van Deursen is a seasoned .NET developer and architect, and the author and maintainer of the Simple Injector DI library. Table of Contents PART 1 Putting Dependency Injection on the map The basics of Dependency Injection: What, why, and how Writing tightly coupled code Writing loosely coupled code PART 2 Catalog DI patterns DI anti-patterns Code smells PART 3 Pure DI Application composition Object lifetime Interception Aspect-Oriented Programming by design Tool-based Aspect-Oriented Programming PART 4 DI Containers DI Container introduction The Autofac DI Container The Simple Injector DI Container The Microsoft.Extensions.DependencyInjection DI Container

Biomedical Natural Language Processing Springer Science & Business Media

Your success—and sanity—are closer at hand when you work at a higher level of abstraction, allowing your attention to be on the business problem rather than the details of the programming platform. Domain Specific Languages—"little languages" implemented on top of conventional programming languages—give you a way to do this because they model the domain of your business problem. DSLs in Action introduces the concepts and definitions a developer needs to build high-quality domain specific languages. It provides a solid foundation to the usage as well as implementation aspects of a DSL, focusing on the necessity of applications speaking the language of the domain. After reading this book, a programmer will be able to design APIs that make better domain models. For experienced developers, the book addresses the intricacies of domain language design without the pain of writing parsers by hand. The book discusses DSL usage and implementations in the real world based on a suite of JVM languages like Java, Ruby, Scala, and Groovy. It contains code snippets that implement real world DSL designs and discusses the pros and cons of each implementation. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book. What's Inside Tested, real-world examples How to find the right level of abstraction Using language features to build internal DSLs Designing parser/combinator-based little languages

Groovy for Domain-specific Languages Simon and Schuster

The success of product line engineering techniques in the last 15 years has popularized the use of software variability as a key modeling approach for describing the commonality and variability of systems at all stages of the software lifecycle. Software product lines enable a family of products to share a common core platform, while allowing for product specific functionality being built on top of the platform. Many companies have exploited the concept of software product lines to increase the resources that focus on highly differentiating functionality and thus improve their competitiveness with higher quality and reusable products and decreasing the time-to-market condition. Many books on product line engineering either introduce specific product line techniques or include brief summaries of industrial cases. From these sources, it is difficult to gain a comprehensive understanding of the various dimensions and aspects of software variability. Here the editors address this gap by providing a comprehensive reference on the notion of variability modeling in the context of software product line engineering, presenting an overview of the techniques proposed for variability modeling and giving a detailed perspective on software variability management. Their book is organized in four main parts, which guide the reader through the various aspects and dimensions of software variability. Part 1 which is mostly written by the editors themselves introduces the major topics related to software variability modeling, thus providing a multi-faceted view of both technological and management issues. Next, part 2 of the book comprises four separate chapters dedicated to research and commercial tools. Part 3 then continues with the most practical viewpoint of the book presenting three different industry cases on how variability is managed in real industry projects. Finally, part 4 concludes the book and encompasses six different chapters on emerging research topics in software variability like e.g. service-oriented or dynamic software product lines, or variability and aspect orientation. Each chapter briefly summarizes “What you will learn in this chapter”, so both expert and novice readers can easily locate the topics dealt with. Overall, the book captures the current state of the art and best practices, and indicates important open research challenges as well as possible pitfalls. Thus it serves as a reference for researchers and practitioners in software variability management, allowing them to develop the next set of solutions, techniques and methods in this complicated and yet fascinating field of software engineering.

Domain-specific Languages Packt Publishing Ltd

The development of modern complex software-intensive systems often involves the use of multiple DSMLs that capture different system aspects. Supporting coordinated use of DSMLs leads to what we call the globalization of modeling languages, that is, the use of multiple modeling languages to support coordinated development of diverse aspects of a system. In this book, a number of articles describe the vision and the way globalized DSMLs currently assist integrated DSML support teams working on systems that span many domains and concerns to determine how their work on a particular aspect influences work on other aspects. Globalized DSMLs offer support for communicating relevant information, and for coordinating development activities and associated technologies within and across teams, in addition to providing support for imposing control over development artifacts produced by multiple teams. DSMLs can be used to support socio-technical coordination by providing the means for stakeholders to bridge the gap between how they perceive a problem and its solution, and the programming technologies used to implement a solution. They also support coordination of work across multiple teams. DSMLs developed in an independent manner to meet the specific needs of domain experts have an associated framework that regulates interactions needed to support collaboration and work coordination across different system domains. The articles in the book describe how multiple heterogeneous modeling languages (or DSMLs) can be related to determine how different aspects of a system influence each other. The book includes a research roadmap that broadens the current DSML research focus beyond the development of independent DSMLs to one that provides support for globalized DSMLs.

Systems and Software Variability Management Packt Publishing Ltd

This book constitutes the refereed proceedings of the 8th International Conference, MLDM 2012, held in Berlin, Germany in

July 2012. The 51 revised full papers presented were carefully reviewed and selected from 212 submissions. The topics range from theoretical topics for classification, clustering, association rule and pattern mining to specific data mining methods for the different multimedia data types such as image mining, text mining, video mining and web mining.

Clojure for Domain-specific Languages Pearson Education

Advances in Computers, Volume 116, presents innovations in computer hardware, software, theory, design, and applications, with this updated volume including new chapters on Teaching Graduate Students How to Review Research Articles and How to Respond to Reviewer Comments, ALGATOR - An Automatic Algorithm Evaluation System, Graph Grammar Induction, Asymmetric Windows in Digital Signal Processing, Intelligent Agents in Games: Review With an Open-Source Tool, Using Clickstream Data to Enhance Reverse Engineering of Web Applications, and more. Contains novel subject matter that is relevant to computer science Includes the expertise of contributing authors Presents an easy to comprehend writing style

Domain-Specific Modeling Createspace Independent Pub

Extend and enhance your Java applications with domain-specific scripting in GroovyAbout This Book- Build domain-specific mini languages in Groovy that integrate seamlessly with your Java apps with this hands-on guide- Increase stakeholder participation in the development process with domain-specific scripting in Groovy- Get up to speed with the newest features in Groovy using this second edition and integrate Groovy-based DSLs into your existing Java applications.Who This Book Is ForThis book is for Java software developers who have an interest in building domain scripting into their Java applications. No knowledge of Groovy is required, although it will be helpful. This book does not teach Groovy, but quickly introduces the basic ideas of Groovy. An experienced Java developer should have no problems with these and move quickly on to the more involved aspects of creating DSLs with Groovy. No experience of creating a DSL is required.What You Will Learn- Familiarize yourself with Groovy scripting and work with Groovy closures- Use the meta-programming features in Groovy to build mini languages- Employ Groovy mark-up and builders to simplify application development- Familiarize yourself with Groovy mark-up and build your own Groovy builders- Build effective DSLs with operator overloading, command chains, builders, and a host of other Groovy language features- Integrate Groovy with your Java and JVM based applicationsIn DetailThe times when developing on the JVM meant you were a Java programmer have long passed. The JVM is now firmly established as a polyglot development environment with many projects opting for alternative development languages to Java such as Groovy, Scala, Clojure, and JRuby. In this pantheon of development languages, Groovy stands out for its excellent DSL enabling features which allows it to be manipulated to produce mini languages that are tailored to a project's needs.A comprehensive tutorial on designing and developing mini Groovy based Domain Specific Languages, this book will guide you through the development of several mini DSLs that will help you gain all the skills needed to develop your own Groovy based DSLs with confidence and ease.Starting with the bare basics, this book will focus on how Groovy can be used to construct domain specific mini languages, and will go through the more complex meta-programming features of Groovy, including using the Abstract Syntax Tree (AST). Practical examples are used throughout this book to de-mystify these seemingly complex language features and to show how they can be used to create simple and elegant DSLs. Packed with examples, including several fully worked DSLs, this book will serve as a springboard for developing your own DSLs.Style and approachThis book is a hands-on guide that will walk you through examples for building DSLs with Groovy rather than just talking about "metaprogramming with Groovy". The examples in this book have been designed to help you gain a good working knowledge of the techniques involved and apply these to producing your own Groovy based DSLs.

Implementing Domain-Specific Languages with Xtext and Xtend Packt Pub Limited

An example-oriented approach to develop custom domain-specific languages.If you've already developed a few Clojure applications and wish to expand your knowledge on Clojure or domain-specific languages in general, then this book is for you. If you're an absolute Clojure beginner, then you may only find the detailed examples of the core Clojure components of value. If you've developed DSLs in other languages, this Lisp and Java-based book might surprise you with the power of Clojure.

DSLs in Action Pearson Education

This book, complete with online files and updates, covers a hugely important area of study in computing. It constitutes the refereed proceedings of the 10th International Symposium on Practical Aspects of Declarative Languages, PADL 2008, held in San Francisco, CA, USA, in January 2008. The 20 revised full papers along with the abstract of 1 invited talk were carefully reviewed and selected from 44 submissions. The papers address all current aspects of declarative programming.

Domain-Specific Languages in Practice John Benjamins Publishing Company

The definitive resource on domain-specific languages: based on years of real-world experience, relying on modern language workbenches and full of examples. Domain-Specific Languages are programming languages specialized for a particular application domain. By incorporating knowledge about that domain, DSLs can lead to more concise and more analyzable programs, better code quality and increased development speed. This book provides a thorough introduction to DSL, relying on today's state of the art language workbenches. The book has four parts: introduction, DSL design, DSL implementation as well as the role of DSLs in various aspects of software engineering. Part I Introduction: This part introduces DSLs in general and discusses their advantages and drawbacks. It also defines important terms and concepts and introduces the case studies used in the most of the remainder of the book. Part II DSL Design: This part discusses the design of DSLs - independent of implementation techniques. It reviews seven design dimensions, explains a number of reusable language paradigms and points out a number of process-related issues. Part III DSL Implementation: This part provides details about the implementation of DSLs with lots of code. It uses three state-of-the-art but quite different language workbenches: JetBrains MPS, Eclipse Xtext and TU Delft's Spoofax. Part IV DSLs and Software Engineering: This part discusses the use of DSLs for requirements, architecture, implementation and product line engineering, as well as their roles as a developer utility and for implementing business logic. The book is available as a printed version (the one your are looking at) and as a PDF. For details see the book's companion website at http: //dslbook.org

Clojure in Action IEEE

Learn to build configuration file readers, data readers, model-driven code generators, source-to-source translators, source analyzers, and interpreters. You don't need a background in computer science--ANTLR creator Terence Parr demystifies language implementation by breaking it down into the most common design patterns. Pattern by pattern, you'll learn the key skills you need to implement your own computer languages.

Knowing how to create domain-specific languages (DSLs) can give you a huge productivity boost. Instead of writing code in a general-purpose programming language, you can first build a custom language tailored to make you efficient in a particular domain. The key is understanding the common patterns found across language implementations. Language Design Patterns identifies and condenses the most common design patterns, providing sample implementations of each. The pattern implementations use Java, but the patterns themselves are completely general. Some of the implementations use the well-known ANTLR parser generator, so readers will find this book an excellent source of ANTLR examples as well. But this book will benefit anyone interested in implementing languages, regardless of their tool of choice. Other language implementation books focus on compilers, which you rarely need in your daily life. Instead, Language Design Patterns shows you patterns you can use for all kinds of language applications. You'll learn to create configuration file readers, data readers, model-driven code generators, source-to-source translators, source analyzers, and interpreters. Each chapter groups related design patterns and, in each pattern, you'll get hands-on experience by building a complete sample implementation. By the time you finish the book, you'll know how to solve most common language implementation problems.

Groovy for Domain-specific Languages Simon and Schuster A general-purpose language like C# is designed to handle all programming tasks. By contrast, the structure and syntax of a Domain-Specific Language are designed to match a particular applications area. A DSL is designed for readability and easy programming of repeating problems. Using the innovative Boo language, it's a breeze to create a DSL for your application domain that works on .NET and does not sacrifice performance. DSLs in Boo shows you how to design, extend, and evolve DSLs for .NET by focusing on approaches and patterns. You learn to define an app in terms that match the domain, and to use Boo to build DSLs that generate efficient executables. And you won't deal with the awkward XML-laden syntax many DSLs require. The book concentrates on writing internal (textual) DSLs that allow easy extensibility of the application and framework. And if you don't know Boo, don't worry-you'll learn right here all the techniques you need. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book.

Advances in Computers Springer Science & Business Media Dijkstra once wrote that computer science is no more about computers than astronomy is about telescopes. Despite the many incredible advances in c- puter science from times that predate practical mechanical computing, there is still a myriad of fundamental questions in understanding the interface between computers and the rest of the world. Why is it still hard to mechanize many tasks that seem to be fundamentally routine, even as we see ever-increasing - pacity for raw mechanical computing? The disciplined study of domain-speci?c languages (DSLs) is an emerging area in computer science, and is one which has the potential to revolutionize the ?eld, and bring us closer to answering this question. DSLs are formalisms that have four general characteristics. – They relate to a well-de?ned domain of discourse, be it controlling tra?c lights or space ships. – They have well-de?ned notation, such as the ones that exist for prescribing music, dance routines, or strategy in a football game. – The informal or intuitive meaning of the notation is clear. This can easily be overlooked, especially since intuitive meaning can be expressed by many di?erent notations that may be received very di?erently by users. – The formal meaning is clear and mechanizable, as is, hopefully, the case for the instructions we give to our bank or to a merchant online.