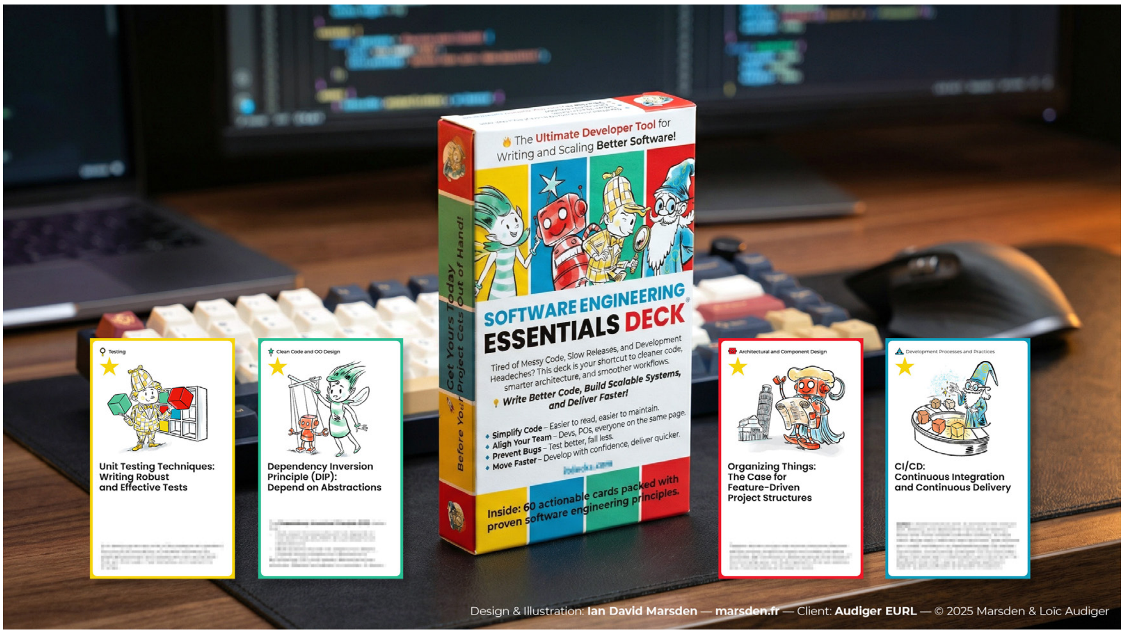

The Ultimate Developer Tool for Writing and Scaling Better Software!
SOFTWARE ENGINEERING ESSENTIALS DECK
Tired of Messy Code, Slow Releases, and Development Headaches? This deck is your shortcut to cleaner code, smarter architecture, and smoother workflows.
Write Better Code, Build Scalable Systems, and Deliver Faster!
Simplify Code – Easier to read, easier to maintain.
Align Your Team – Devs, POs, everyone on the same page.
Prevent Bugs – Test better, fail less.
Move Faster – Develop with confidence, deliver quicker.
Inside: 60 actionable cards packed with proven software engineering principles.
Before Your Project Gets Out of Hand! Get Yours Today
Testing
Unit Testing Techniques: Writing Robust and Effective Tests
Clean Code and OO Design
Dependency Inversion Principle (DIP): Depend on Abstractions
Architectural and Component Design
Organizing Things: The Case for Feature-Driven Project Structures
Development Processes and Practices
CI/CD: Continuous Integration and Continuous Delivery

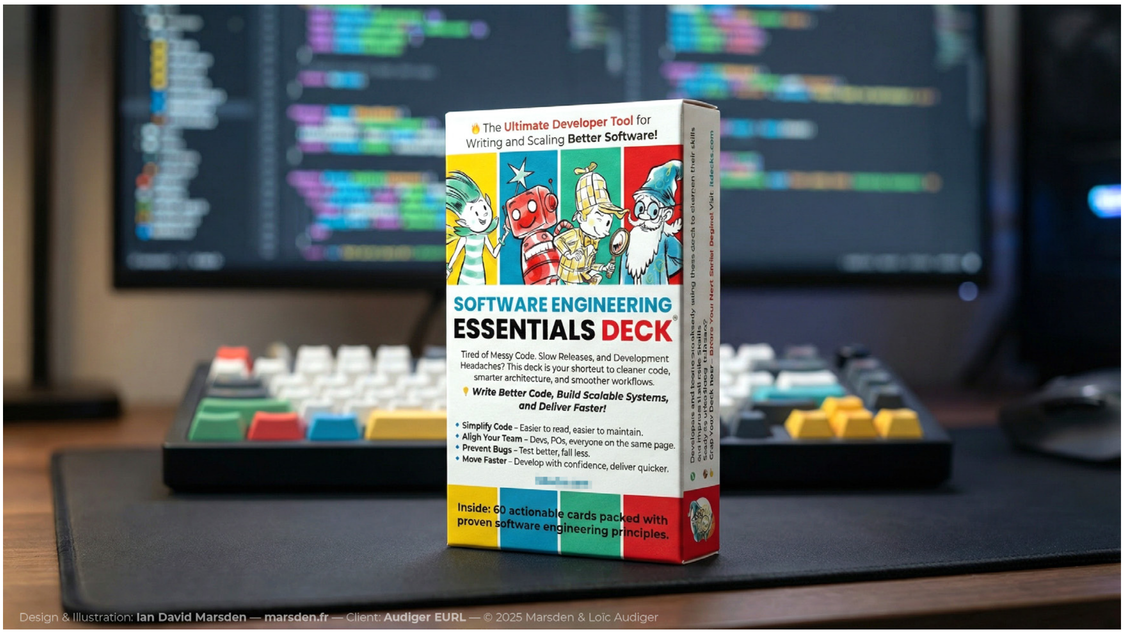
Design & Illustration: **Ian David Marsden** — **marsden.fr** — Client: **Audiger EURL** — © 2025 Marsden & Loic Audiger

The Ultimate Developer Tool for Writing and Scaling Better Software!
SOFTWARE ENGINEERING ESSENTIALS DECK
Inside: 60 actionable cards packed with proven software engineering principles.
Unit Testing Techniques: Writing Robust and Effective Tests
Organizing Things: The Case for Feature-Driven Project Structures
Dependency Inversion Principle (DIP): Depend on Abstractions

Design & Illustration: **Ian David Marsden** — **marsden.fr** — Client: **Audiger EURL** — © 2025 Marsden & Loic Audiger

**CASE study**

# SOFTWARE ENGINEERING ESSENTIALS DECK©


The Ultimate Developer Tool for Writing and Scaling Better Software!
SOFTWARE ENGINEERING ESSENTIALS DECK
Inside: 60 actionable cards packed with proven software engineering principles.

Design & Illustration: **Ian David Marsden** — **marsden.fr** — Client: **Audiger EURL** — © 2025 Marsden & Loic Audiger

Design by Marsden Creative (2025)

# Ian David Marsden

**Creative Director · Illustrator · Brand Consultant**

Visual Storytelling · UX/UI Design · Motion & Branding · Editorial & Character Illustration

📍 Based in France – Available Worldwide

☎ +33 6 42 49 82 11 · ✉ ian@marsden.fr

🌐 marsden.fr · linkedin.com/in/idmarsden · youtube.com/@MarsdenIllustration · vimeo.com/idmarsden

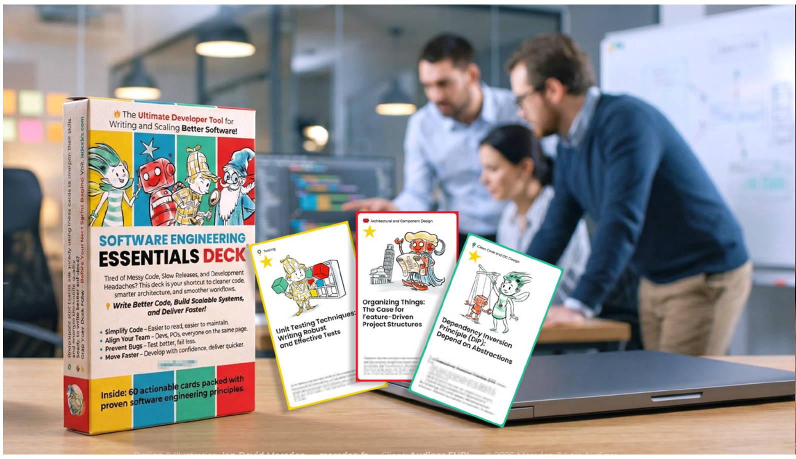
## CASE STUDY - MARSDEN CREATIVE

## Visual System & Character Design for a **Software Engineering Training Deck**

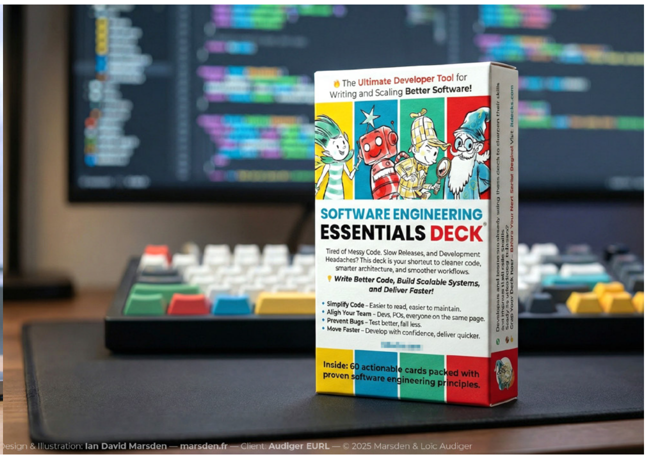
Illustration and layout system for developer training on CI/CD, TDD, and software architecture

Client: **Loïc Audiger**, CTO
Contact: **Audiger EURL**
Location: **Montpellier**, France
Role: **Creative Director · Illustrator · Character Designer · Iconographer · Packaging & Production Lead**
Project Type: **EdTech Product Design / Strategy Deck for Professional Training**

*This project is featured with client permission for portfolio use. Card content and instructional materials remain ©2025 Loïc Audiger, all rights reserved.*



The Ultimate Developer Tool for Writing and Scaling Better Software!
SOFTWARE ENGINEERING ESSENTIALS DECK
Inside: 60 actionable cards packed with proven software engineering principles.
Unit Testing Techniques: Writing Robust and Effective Tests
Organizing Things: The Case for Feature-Driven Project Structures
Dependency Inversion Principle (DIP): Depend on Abstractions



The Ultimate Developer Tool for Writing and Scaling Better Software!
SOFTWARE ENGINEERING ESSENTIALS DECK
Inside: 60 actionable cards packed with proven software engineering principles.

Design & Illustration: Ian David Marsden — marsden.fr — Client: Audiger EURL — © 2025 Marsden & Loïc Audiger

### Project Brief & Visual Problem-Solving

The client supplied a technically robust curriculum: a 59-card strategy deck developed to support software engineering training for teams and individuals. Topics spanned from Domain-Driven Design and Refactoring to CI/CD, TDD, OKRs, and the long-term management of Technical Debt. While the content was sound and clearly organized, it lacked visual cohesion.

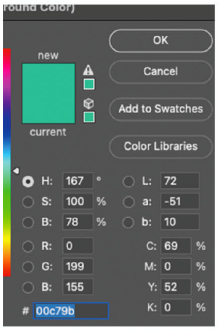
The brief was to create a logic-driven visual framework that could aid comprehension, engagement, and scalability — supporting both independent learning and instructor-led workshops. The solution would need to communicate clearly without oversimplifying the complexity of modern development practices..

### Instructional Taxonomy and Visual System Architecture

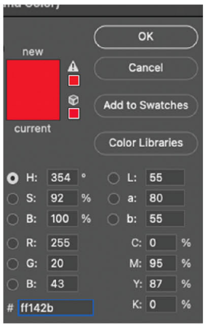
Rather than approaching each card as a one-off illustration, I proposed a quadrant-based structure to serve as the system's foundation. The cards were divided into four core thematic domains, each assigned a symbolic color, character, icon, and layout convention.

## CHARTE COULEURS

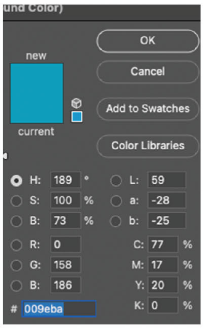
| Code Sprite | Robot | Wizard | Inspector |
|---|---|---|---|
| 00c79b | ff142b | 009eba | ffd400 |



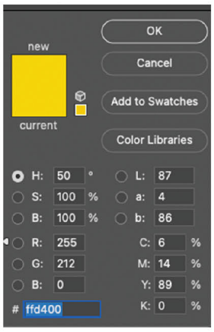
```
● H: 167 °     ○ L: 72
○ S: 100 %     ○ a: -51
○ B: 78  %     ○ b: 10
○ R: 0           C: 69 %
○ G: 199         M: 0  %
○ B: 155         Y: 52 %
# 00c79b         K: 0  %
```



```
● H: 354 °     ○ L: 55
○ S: 92  %     ○ a: 80
○ B: 100 %     ○ b: 55
○ R: 255         C: 0  %
○ G: 20          M: 95 %
○ B: 43          Y: 87 %
# ff142b         K: 0  %
```



```
● H: 189 °     ○ L: 59
○ S: 100 %     ○ a: -28
○ B: 73  %     ○ b: -25
○ R: 0           C: 77 %
○ G: 158         M: 17 %
○ B: 186         Y: 20 %
# 009eba         K: 0  %
```

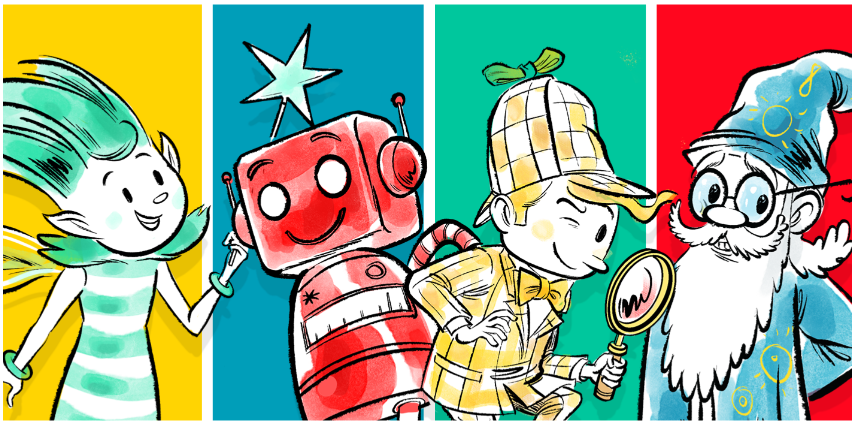


```
● H: 50  °     ○ L: 87
○ S: 100 %     ○ a: 4
○ B: 100 %     ○ b: 86
○ R: 255         C: 6  %
○ G: 212         M: 14 %
○ B: 0           Y: 89 %
# ffd400         K: 0  %
```

Category 01 (Clean Code): Green (Sprite character)
Category 02 (Architecture): Red (Robot character)
Category 03 (Processes): Blue (Wizard character)
Category 04 (Testing): Yellow (Detective character)

- **Green** – **Clean Code & Object-Oriented Design**
  *Covering syntax, logic, and the foundational structures of programming.*
- **Red** – **Architecture & Component Design**
  *Representing modular design principles, clean architecture, and Domain-Driven Design (DDD) patterns.*
- **Blue** – **Development Processes & Team Practices**
  *Focused on agile iteration, refactoring, team coordination, and strategic planning through frameworks like OKRs.*
- **Yellow** – **Testing & Quality Assurance**
  *Dedicated to test logic, review protocols, and automation practices including Test-Driven Development (TDD).*

This color-coded taxonomy provided a consistent visual and conceptual structure — supporting both intuitive navigation and long-term learning retention..



### Character Design: Semantic Anchors

The four illustrated characters were designed not as mascots, but as mnemonic anchors — each marking a conceptual domain while supporting memory, orientation, and visual rhythm across the 59-card system.

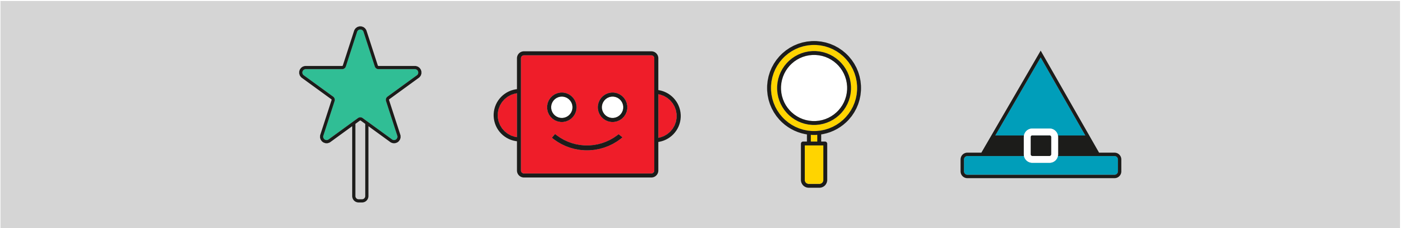
**The Sprite** (Green – Clean Code and Object-Oriented Design) is agile, luminous, and fast-moving—her striped form and winged hair suggest lightness and adaptability. Equipped with a magical wand, she symbolizes the power to refactor, reframe, and clarify code on the fly. Her presence marks cards focused on syntax, modularity, and expressive logic—elegant, readable solutions in motion.

**The Robot** (Red – Architecture and Component Design) draws from classic vintage toy aesthetics: segmented arms, antennae, and visible joints speak to his modular nature. With a steady stance and friendly gaze, he represents structural clarity, encapsulation, and mechanical precision—the foundations of Domain-Driven Design and robust system boundaries.

**The Wizard** (Blue – Development Processes and Team Practices) is modeled on the archetype of Merlin—a figure of wisdom and foresight. He carries schematic scrolls and magical instruments, embodying planning, iteration, and long-view strategy. His role is to guide the learner through refactoring cycles, team rituals, and evolving delivery systems—a visual companion to CI/CD pipelines and process excellence.

**The Inspector** (Yellow – Testing and Quality Assurance) channels the spirit of Sherlock Holmes—complete with deerstalker hat, magnifying glass, and a sharply observant posture. He personifies rigorous examination, unit testing, and quality control. With humor and precision, he reinforces the discipline of catching flaws, validating logic, and reinforcing robustness.

Their visual design was deliberately restrained: symbolic, not theatrical; readable at card scale; and distinct without relying on heavy-handed tropes. Each reinforces its respective color-coded domain and contributes to the system's internal rhythm.



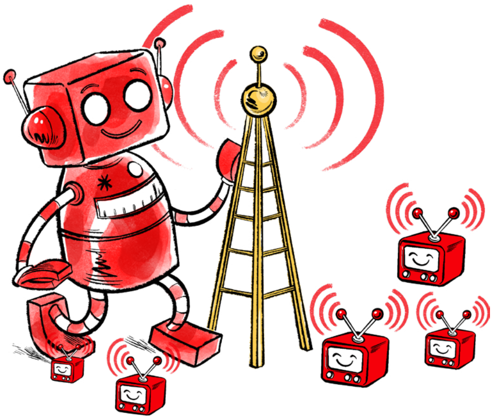
## Icon System and Visual Tagging

Each domain was further supported by a custom-designed **vector icon**: geometric, high-contrast, and scalable. These icons functioned as tags throughout the visual system — appearing on card fronts, documentation, and packaging. Their role was structural: helping learners orient themselves, return to concepts, and mentally cluster related practices.

## Illustration System: 59 Visual Scenes

Each card was illustrated as a distinct instructional vignette — metaphorical, never ornamental. The goal was to trigger recognition, reflection, or dialogue.

Stylistically, the illustrations used clean digital linework with soft tonal coloring. Each adhered to its domain palette (green, red, blue, yellow) to reinforce categorical logic. The tone was intentionally quiet — precise enough for professionals, open enough to support discussion.

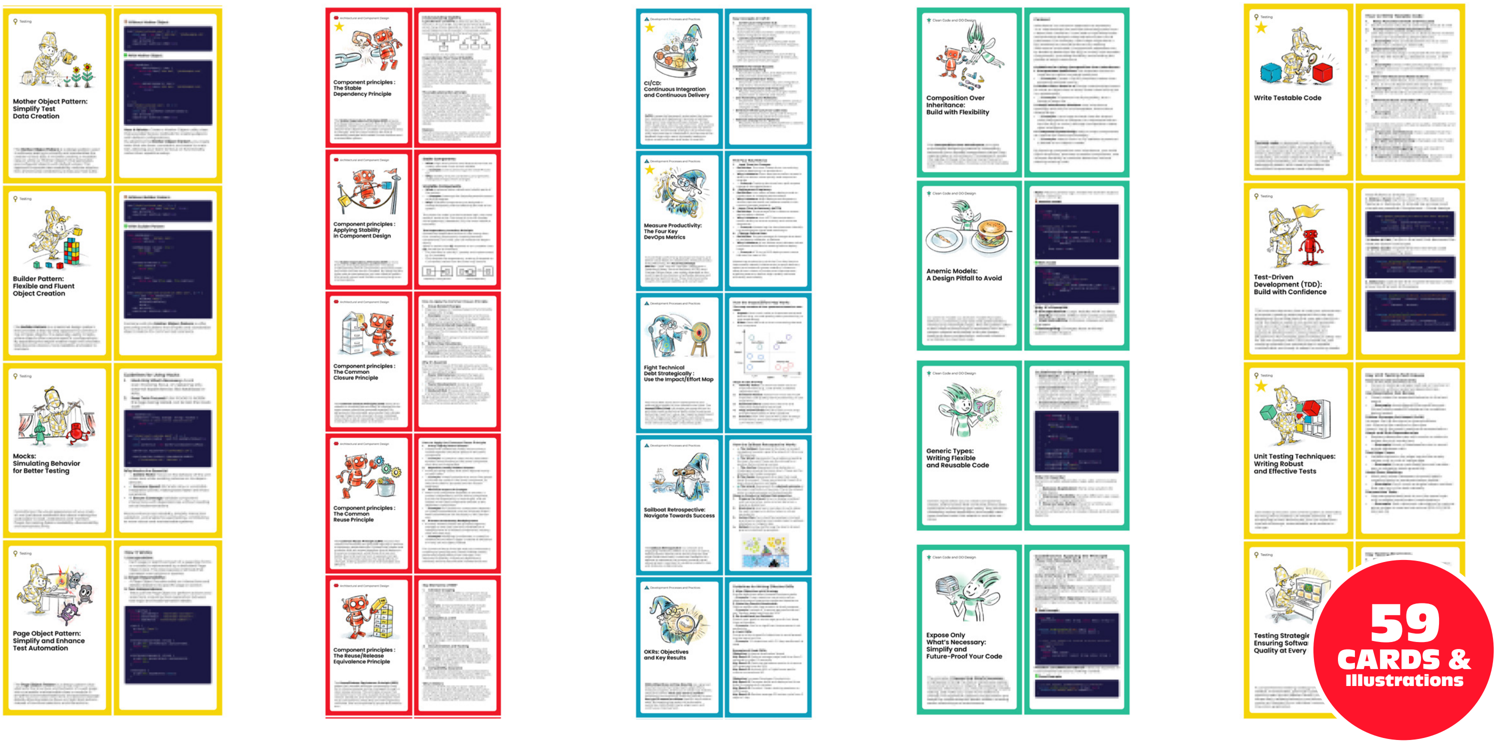


Illustrations ©2025 - Ian David Marsden

## Card Layout and Visual Hierarchy

A scalable card layout system was developed to ensure visual consistency across all 59 cards:

- A modular grid for alignment and rhythm
- Defined image and text zones
- Hierarchical type styles to support varying content density


Mother Object Pattern: Simplify Test Data Creation
Builder Pattern: Flexible and Fluent Object Creation
Mocks: Simulating Behavior for Better Testing
Page Object Pattern: Simplify and Enhance Test Automation
Component principles : The Stable Dependency Principle
Component principles : Applying Stability in Component Design
Component principles : The Common Closure Principle
Component principles : The Common Reuse Principle
Component principles : The Reuse/Release Equivalence Principle
CI/CD: Continuous Integration and Continuous Delivery
Measure Productivity: The Four Key DevOps Metrics
Fight Technical Debt Strategically: Use the Impact/Effort Map
Sailboat Retrospective: Navigate Towards Success
OKRs: Objectives and Key Results
Composition Over Inheritance: Build with Flexibility
Anemic Models: A Design Pitfall to Avoid
Generic Types: Writing Flexible and Reusable Code
Expose Only What's Necessary: Simplify and Future-Proof Your Code
Write Testable Code
Test-Driven Development (TDD): Build with Confidence
Unit Testing Techniques: Writing Robust and Effective Tests
Testing Strategie Ensuring Softwa Quality at Every
59 CARDS & illustrations

Each card in the system was designed to serve a dual function: as an instructional aid and as a navigational tool. The front provided clear orientation — showing the module name, a concise title and description, and a custom illustration aligned with its domain. The back contained proprietary content from the client, including structured text, example code, and domain-specific insights.

This layout supported multiple learning styles: solutions could be explored through visual association, structured reflection, or collaborative brainstorming. It allowed the cards to function as prompts — not just reference material.

The design itself had to remain subtle. It needed to support focus, not demand it. A modular grid ensured visual rhythm across all 59 cards, with clearly defined zones for image, text, and metadata. Readability was prioritized over ornament. The layout system also allows for future cards to be added without disrupting structure — supporting scalability without redesign.

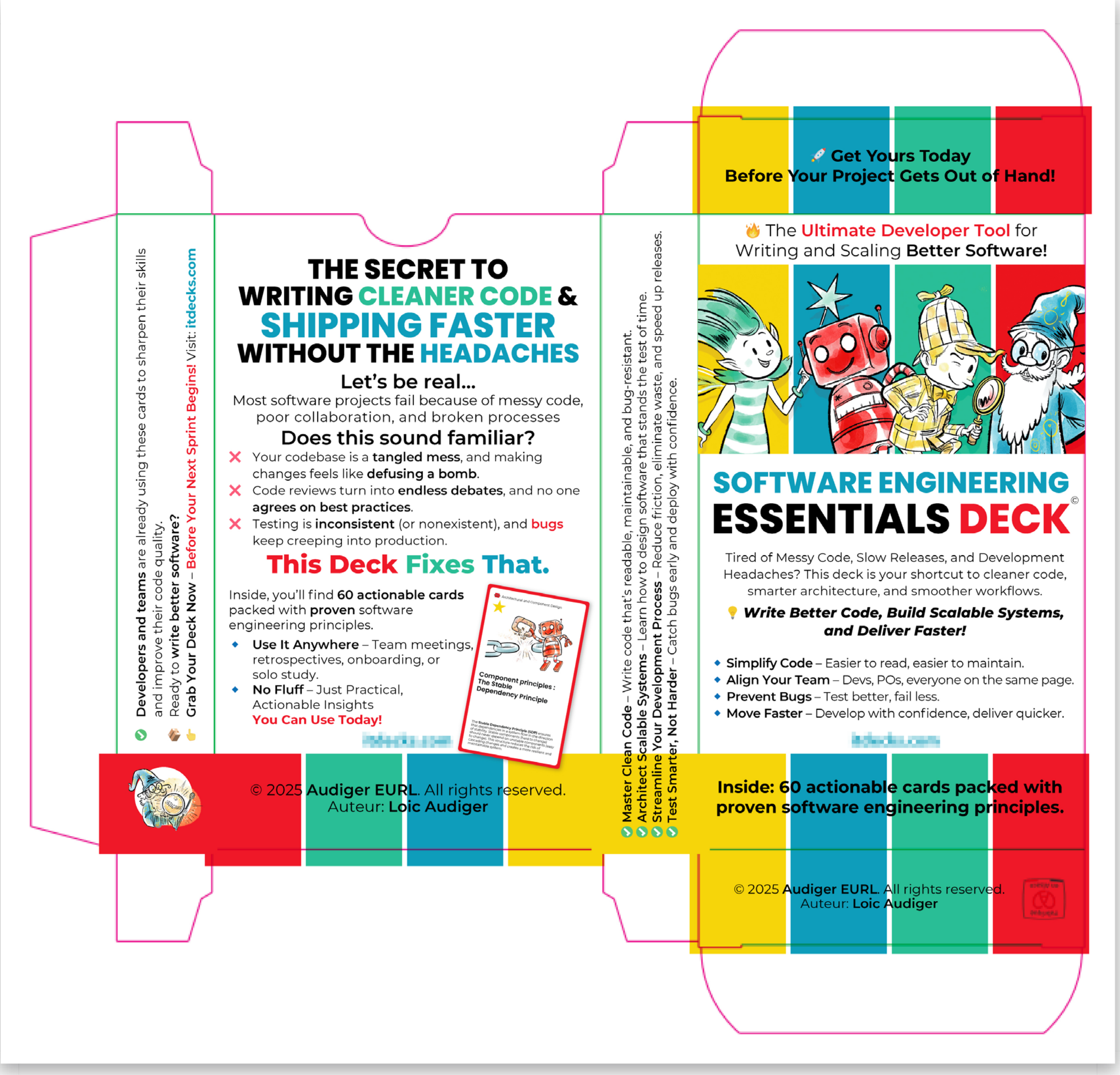
## Packaging and Production Workflow

The cards were housed in a custom-designed étui box. I handled the full packaging development cycle — from dieline to final prepress delivery — and produced the entire project on a flat-rate basis, agreed upfront.

Deliverables included:

- Illustrator dieline file with fold and glue annotations
- Visual alignment between packaging and internal card system
- Final offset print files with CMYK conversion
- Color-managed print assets delivered as PDF/X-4

Usage rights for all artwork and design were transferred to the client upon delivery, with illustrations also supplied as individual assets — prepared for use in secondary applications such as presentations, advertising, and social media.


Developers and teams are already using these cards to sharpen their skills and improve their code quality.
Ready to write better software?
Grab Your Deck Now – Before Your Next Sprint Begins! Visit: itdecks.com
THE SECRET TO WRITING CLEANER CODE & SHIPPING FASTER WITHOUT THE HEADACHES
Let's be real...
Most software projects fail because of messy code, poor collaboration, and broken processes
Does this sound familiar?
✕ Your codebase is a tangled mess, and making changes feels like defusing a bomb.
✕ Code reviews turn into endless debates, and no one agrees on best practices.
✕ Testing is inconsistent (or nonexistent), and bugs keep creeping into production.
This Deck Fixes That.
Inside, you'll find 60 actionable cards packed with proven software engineering principles.
◆ Use It Anywhere – Team meetings, retrospectives, onboarding, or solo study.
◆ No Fluff – Just Practical, Actionable Insights You Can Use Today!
Architectural and Component Design
Component principles : The Stable Dependency Principle
© 2025 Audiger EURL. All rights reserved.
Auteur: Loic Audiger
Master Clean Code – Write code that's readable, maintainable, and bug-resistant.
Architect Scalable Systems – Learn how to design software that stands the test of time.
Streamline Your Development Process – Reduce friction, eliminate waste, and speed up releases.
Test Smarter, Not Harder – Catch bugs early and deploy with confidence.
Get Yours Today
Before Your Project Gets Out of Hand!
The Ultimate Developer Tool for Writing and Scaling Better Software!
SOFTWARE ENGINEERING ESSENTIALS DECK©
Tired of Messy Code, Slow Releases, and Development Headaches? This deck is your shortcut to cleaner code, smarter architecture, and smoother workflows.
Write Better Code, Build Scalable Systems, and Deliver Faster!
◆ Simplify Code – Easier to read, easier to maintain.
◆ Align Your Team – Devs, POs, everyone on the same page.
◆ Prevent Bugs – Test better, fail less.
◆ Move Faster – Develop with confidence, deliver quicker.
Inside: 60 actionable cards packed with proven software engineering principles.
© 2025 Audiger EURL. All rights reserved.
Auteur: Loic Audiger

## Final Deliverables

The complete system was delivered as a production-ready, modular asset library, including:

- Adobe Illustrator (AI) source files
- Print-ready PDF/X-4 files
- SVG/PDF icon sets
- High-resolution PNG/JPEG exports
- Typography and color specifications
- Individual illustrations for reuse in promotional materials
- Full usage rights transferred to the client
- Layout and visual system structured for future expansion

## Outcome

The final result is a cohesive instructional product — one that translates complex software engineering concepts into a structured visual system. The illustrations, layout, characters, and packaging all work together to support understanding, recall, and modular delivery.

This was a project grounded in alignment — between narrative logic, technical structure, and instructional UX. The system clarifies without simplifying, and structures without rigidity — a design approach that reflects the same thinking developers apply in code.

Design by **Marsden Creative** (2025)
marsdenillustration.com
contact@marsdenillustration.com