
Upgrading Legacy Code SE Radio 602

- Pablo Pérez Álvarez
- Bruno Pérez Cuervo
- Daniel Flores López

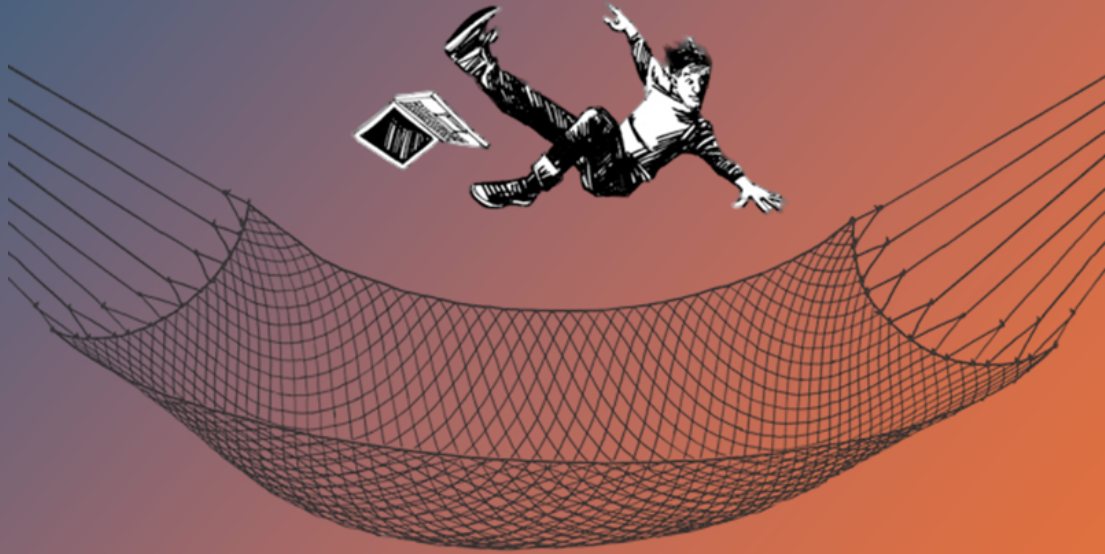


Legacy Code

¿Qué es?



¿Reescribir o refactorizar?



Safety nets

- Tests automáticos
- Recuperar documentación antigua
- Recuperar conocimiento antiguo
- Monitorear logs

SAAQclic a total failure, hundreds of millions over budget, auditor general finds

Decision-makers knew system wasn't ready but launched it anyways



[Matthew Lapierre, Isabelle Alexandre](#) · CBC News ·

Posted: Feb 20, 2025 12:17 PM EST | Last Updated: February 21





¿Las empresas no quieren refactorizar?

Primera razón:
Falta de
comunicación
apropiada



Segunda razón:
Personas
arraigadas a una
manera de
trabajar





No centrarse
en las partes
más complejas



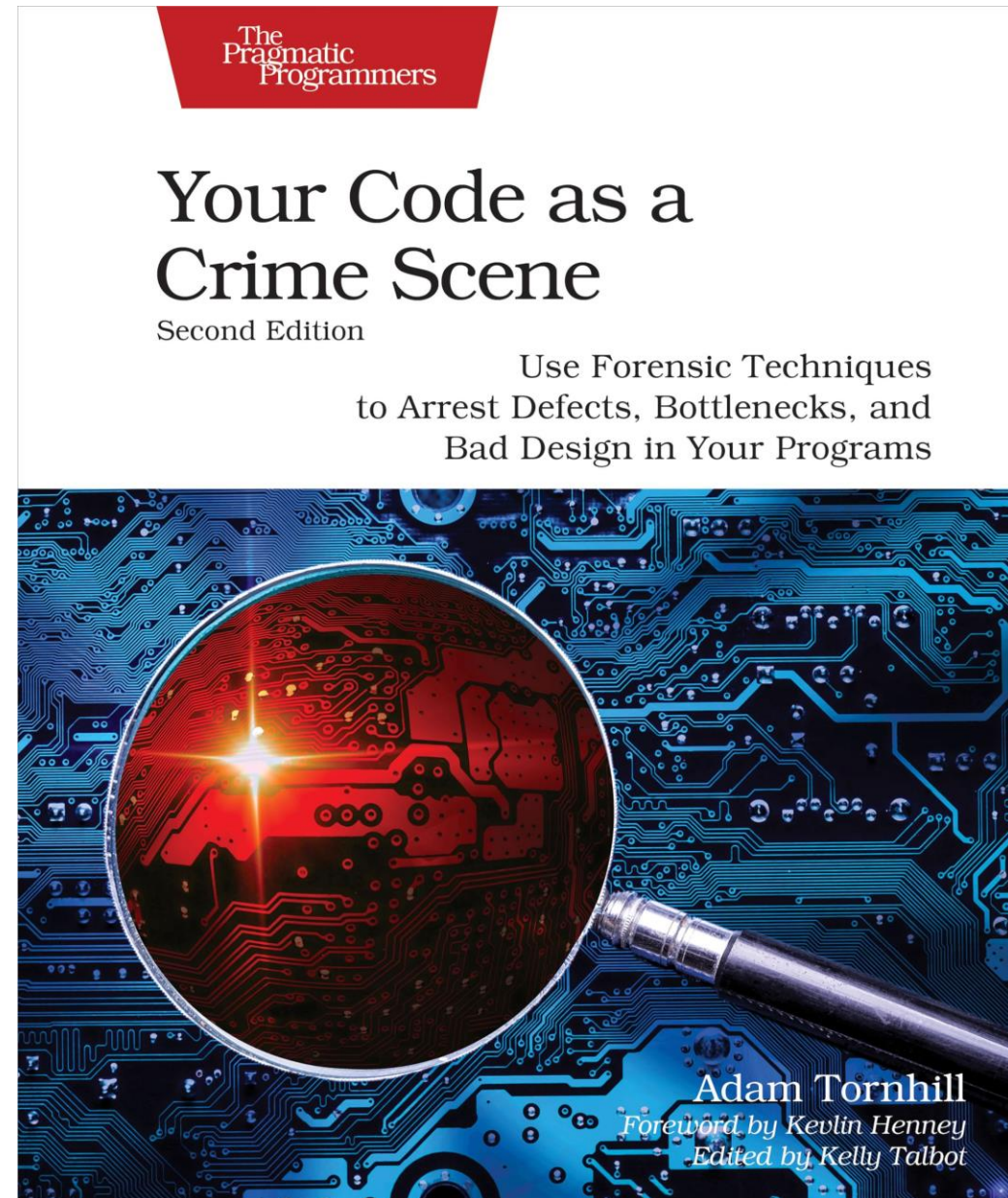


Predecir lo justo y necesario



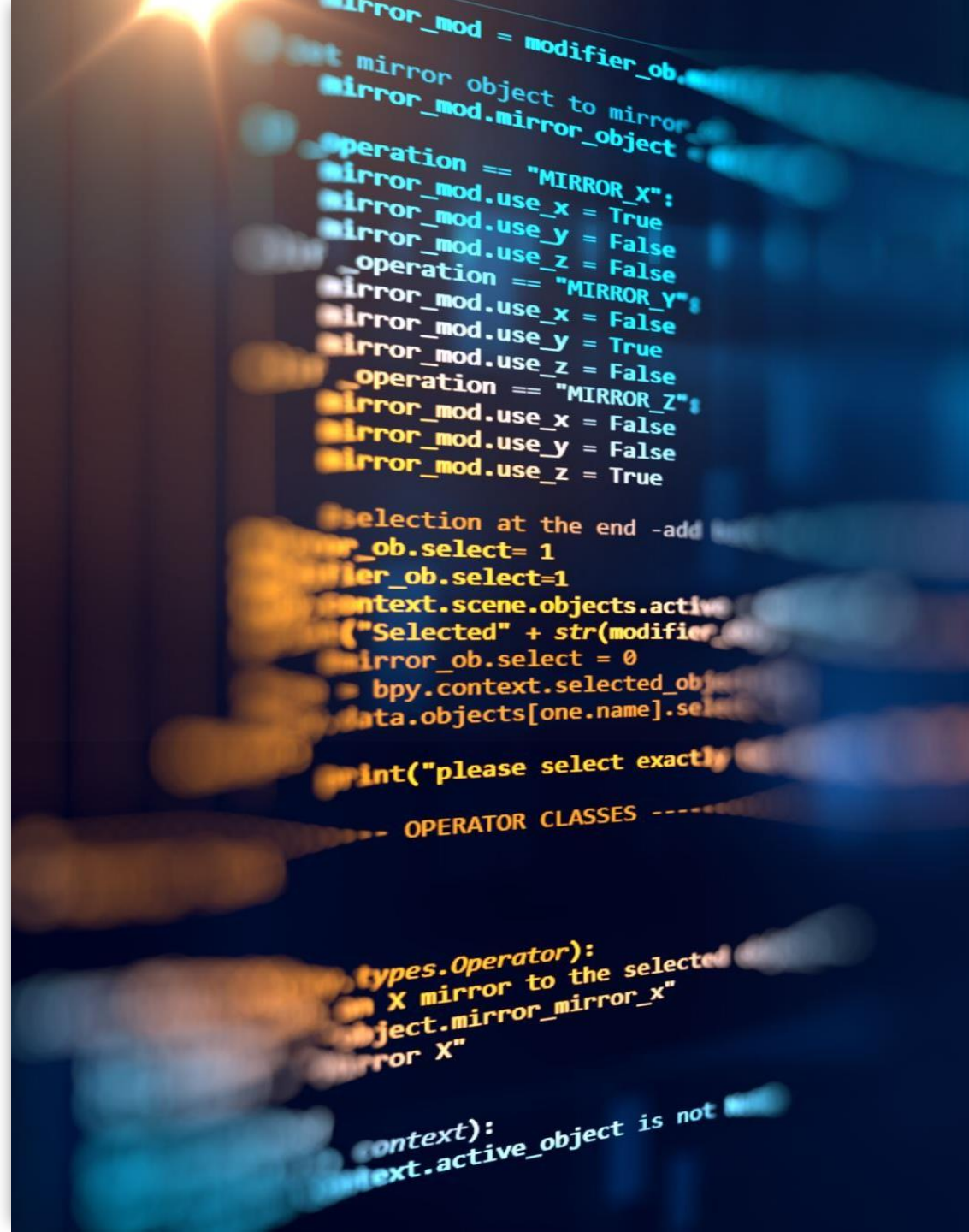
Refactorizar como parte del desarrollo

Tu código
como una
escena del
crimen



Resumen del libro

- Compara el código de desarrolladores que abandonaron la empresa con una escena del crimen.
- Puedes tomar el papel de un examinador forense con el código, usando herramientas como git para ver quién escribió que parte del código, quien era un experto en que parte, etc..
- Puedes juntar esto con otras herramientas de análisis para analizar tu propio código, para ver, por ejemplo, que archivos se modifican más.



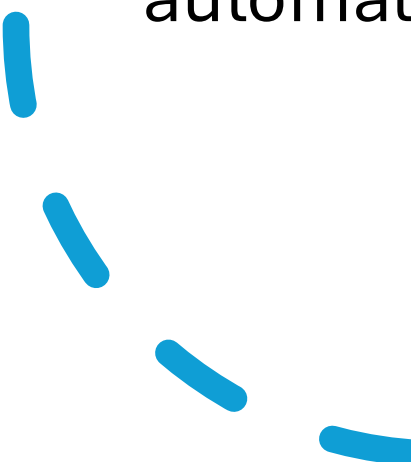
Invertir en la limpieza del código

- Partiendo de lo que se acaba de hablar, podemos hacer un "Hotspot analysis".
- Esto trata de analizar, usando datos de control de versiones en que partes del código vale más la pena invertir el tiempo, para maximizar la eficiencia, lo que podría considerarse una inversión.



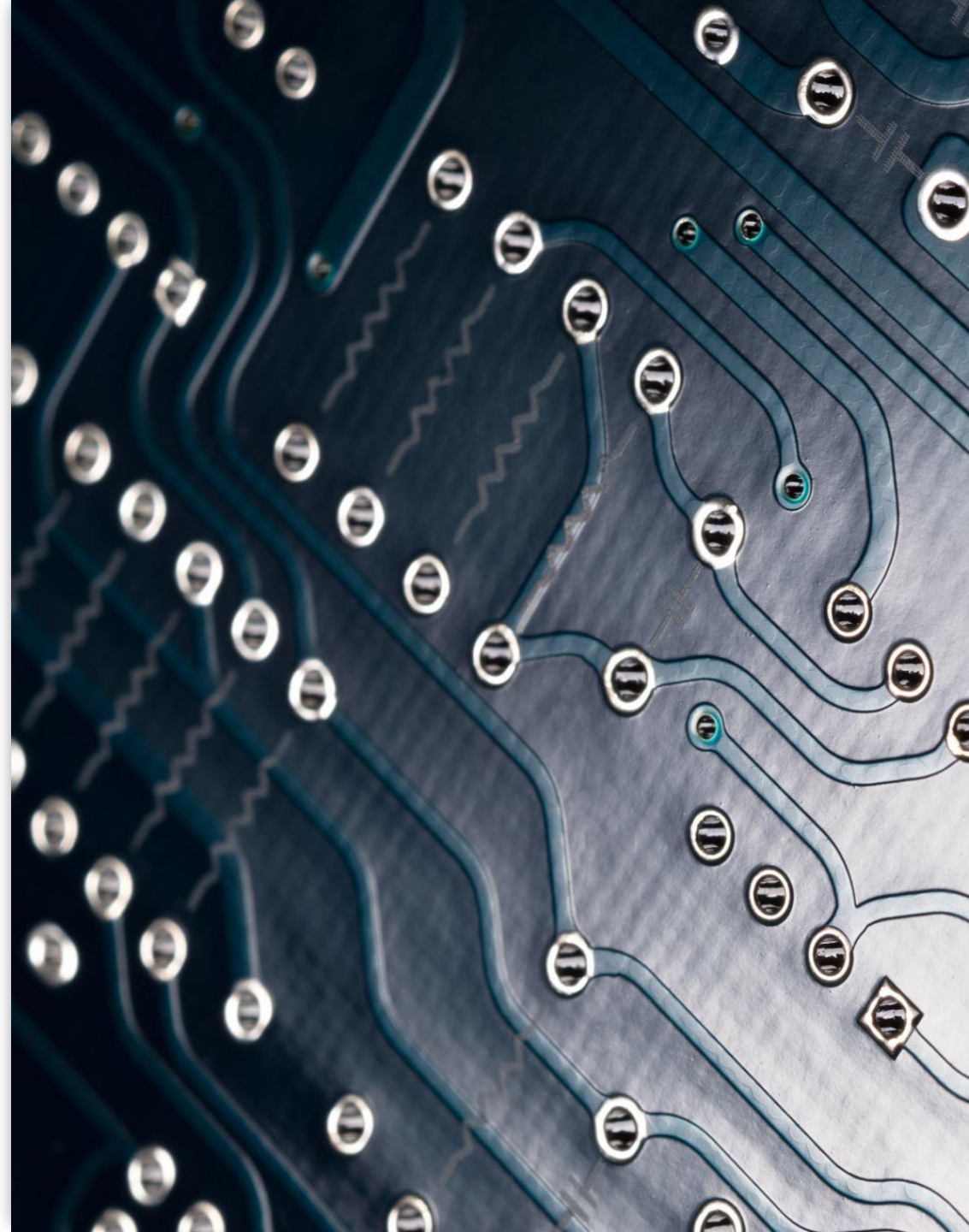


¿Qué técnicas del libro son las más comunes?

- Nicolas habla de que lo primero que hace al llegar a un nuevo proyecto, en caso de ser posible, es un hotspot analysis.
 - Luego, dentro del propio código suele usar bastante las técnicas de refactorización, siendo la más frecuente de ellas la separación de la capa de negocio y lógica, además de la refactorización automática que traen los editores de código.
- 

La inteligencia artificial en el legacy code

- La inteligencia artificial puede ser una herramienta útil a la hora de tratar con legacy code, esta nos puede ayudar de distintas maneras, como, por ejemplo, le podemos pasar una función compleja de la que no entendemos su funcionamiento .
- Otra forma de la que nos puede ayudar, y que algunas empresas están haciendo ya, es a desarrollar los tests puramente con inteligencia artificial.
- Y finalmente, también la podemos usar para refactorizar un fragmento de código.





Problemas de usar la inteligencia artificial

- **Hay tres principales problemas:**
 - Saber si falta algún test.
 - Saber si la calidad de los tests es suficiente.
 - A la hora de refactorizar código, saber que este haga lo mismo que hacía antes.
- Nicolas recomienda escribir un par de casos de test a mano y luego ya podemos usar una inteligencia artificial para ayudarnos a hacer pruebas más complejas, o expandirlas, una vez tengamos unas pruebas robustas podríamos usar la inteligencia artificial para refactorizar el código.

Herramientas para análisis de código

- Según Nicolas Carlo, es un concepto bastante nuevo en los desarrolladores.
- También habla del análisis de comportamiento, relacionado con Adam Tornhill
- Adam Tornhill tiene un SaaS, CodeScene
- Gratis en openSources, de pago en proyectos privados.

VISUALIZE CODE HEALTH with CodeScene's File-level Hotspots


By default, CodeScene calculates the code health of all the files in your code base

CodeScene


Hotspots Refactoring Targets Costs Defects Programming Language **Code Health**


System / mongo / src / mongo / db / repl / replication_coordinator_impl.cpp


 Healthy Problematic Unhealthy ...

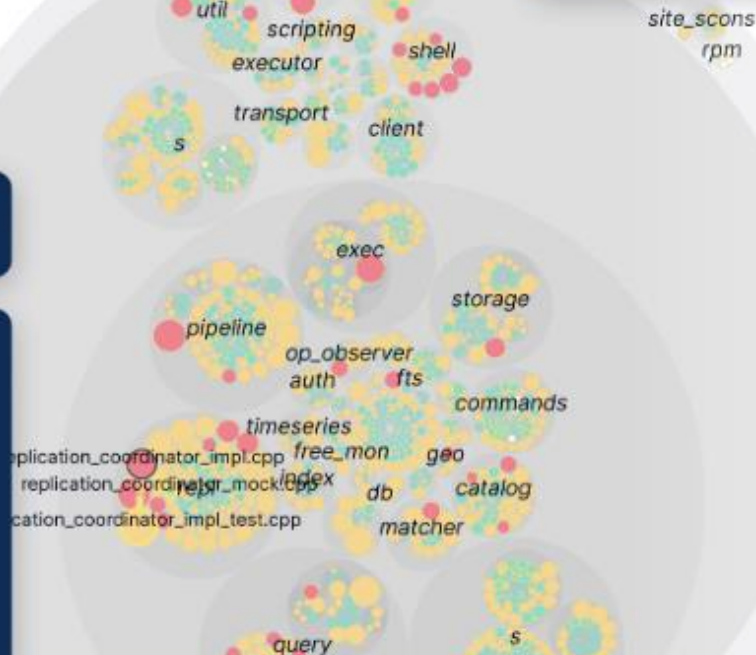
 Zoom in to specific files in your codebase

 CodeScene classifies all your code into three categories

 Green code - healthy code that is easy for developers to pick up, understand and evolve

 Yellow code - this is where you've started to take on technical debt

 Red code - code that's expensive and challenging to maintain



 Search by filename

 Filter by owner

Code Health Range

Commit Threshold 

 /  replication_coordinator_impl.cpp

Review 

Metrics Complexity Trends

 Code Health

Unhealthy

Commits 66 commits / 1

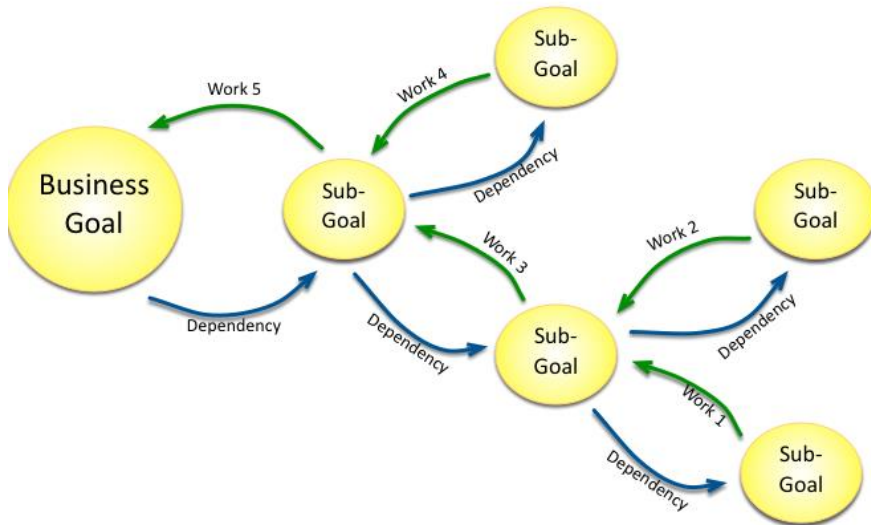
Size 4,692 Lines of Code

Main Author Spencer T. Brown

Knowledge 75 % code by 1 contributors

- Cuantifica y prioriza la deuda técnica
- Revisiones automatizadas de código
- Extensiones con feedback de Code Health
- Se centra en lo fácil que sea el código de mantener, leer o cambiar

Método Mikado



- Iteración base:
 - Se marca un objetivo
 - Se intenta solucionar en 10-15 minutos (lo suyo es no conseguirlo)
 - Se concluye en dividir en subtareas y encontrar bloqueos
- Siguiendo iteración:
 - Se escoge una subtask de la que depende el objetivo
 - Se desecha el código escrito hasta ahora
 - Intentar realizar la subtask entre 10-15 minutos
- Si no se consigue, se escribe la información encontrada y se divide en subtareas.

Método Mikado

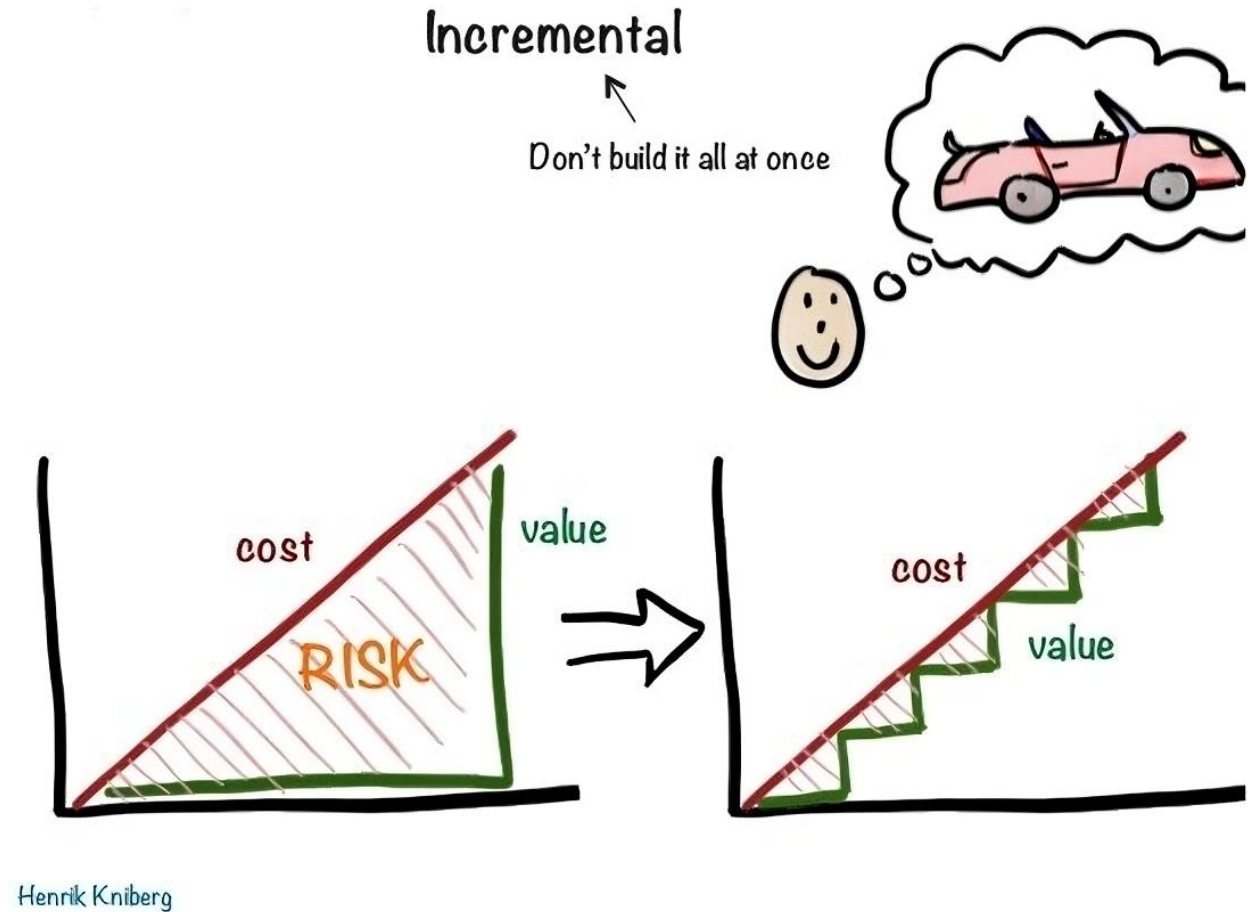
- Nicolas Carlo la recomienda en tareas grandes o cuando se es nuevo en un proyecto
- Beneficios:
 - Se habrán resuelto la mayoría de bloqueos para realizar la tarea
 - Se tendrá una lista ordenada de qué hacer
 - Se habrá generado un artefacto visual que se puede usar para aclarar ideas y debatir tu visión con tus compañeros.

Hábitos a mejorar en los desarrolladores

- Nicolas Carlo enfatiza en lo que él llama "micro committing"
- Los commits deben ser pequeños y con un enfoque común.
- Entiende la dificultad de esto porque Nicolas también tenía problemas con ello.
- Consejos:
 - Usar Mikado, al final divide en subtareas
 - Refactoring incrementales
 - Recordatorios sonoros o alarmas cada 5 minutos (bajo su experiencia)

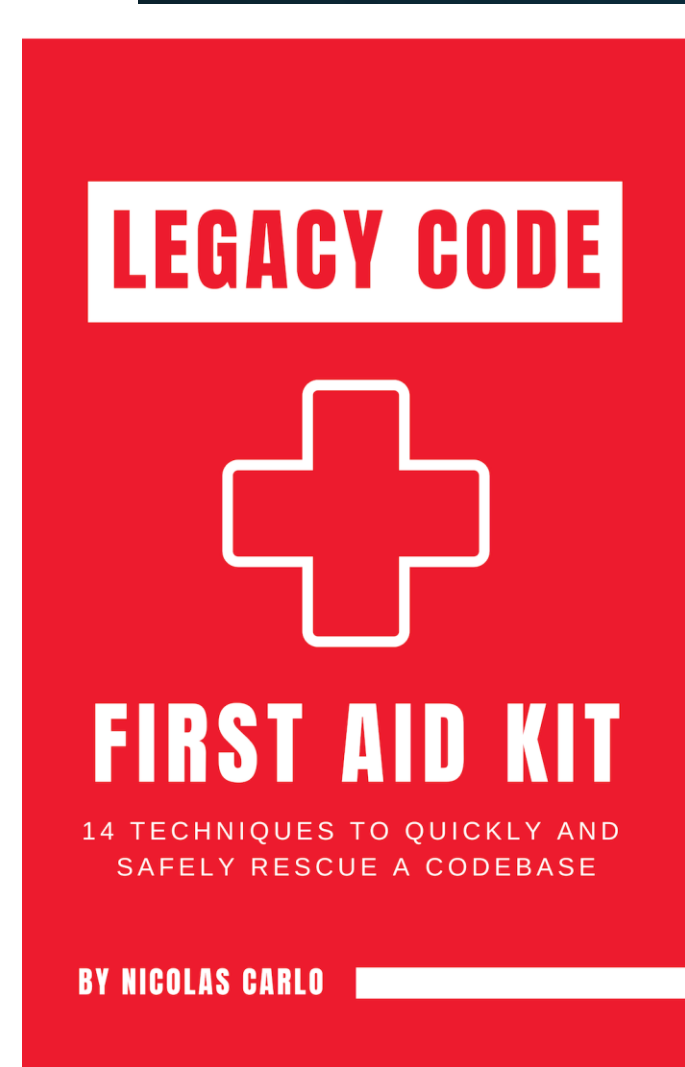
Refactoring incrementales

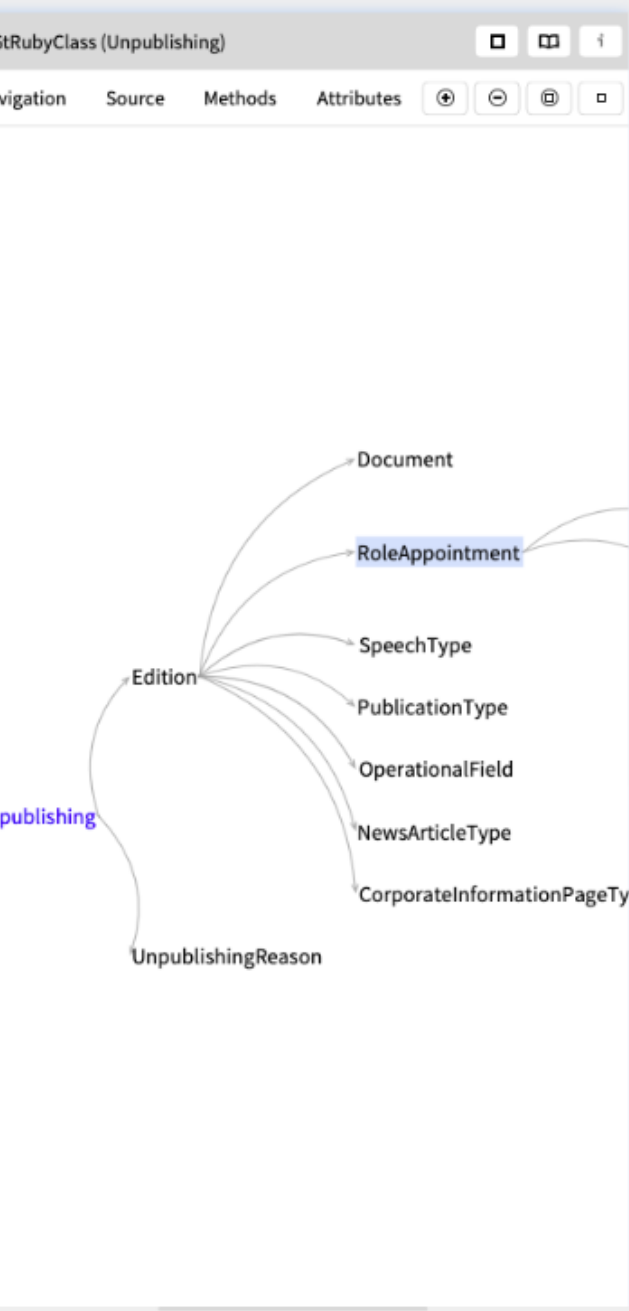
- Muy acorde a las metodologías ágiles
- Cambios graduales en la aplicación hasta llegar al objetivo
- Sumar funcionalidades pero no todas a la vez



Aprendizaje post-libro

- La mayoría de técnicas siguen siendo muy válidas dada la naturaleza del problema
- La IA es un fenómeno que no era así hace 2-3 años
- Podría combinar y reforzar algunas de las técnicas propuestas
- Ve que el approval testing es más interesante de lo que pensaba.





a GtRubyClass (RoleAppointment)

```

class RoleAppointment < ApplicationRecord
  include HasContentId
  include PublishesToPublishingApi

  CURRENT_CONDITION = { ended_at: nil }.freeze

  has_many :edition_role_appointments
  has_many :editions,
  through: :edition_role_appointments

  # All this nonsense is because of all the
  intermediary associations
  has_many :consultations,
    -> { where("editions.type" =>
  "Consultation") },
  through: :edition_role_appointments,
  source: :edition
  has_many :publications,
    -> { where("editions.type" =>
  "Publication") },
  through: :edition_role_appointments,
  source: :edition
  has_many :news_articles,
    -> { where("editions.type" =>
  "NewsArticle") },
  through: :edition_role_appointments,
  source: :edition

  # Speeches do not need the above nonsense because
  they have a singular
  # association in the 'editions' table
  has_many :speeches

  belongs_to :role
  belongs_to :person
  has_many :organisations, through: :role

  delegate :slug, to: :person
  delegate :name, to: :role, prefix: true
  delegate :ministerial?, to: :role

  class Validator < ActiveModel::Validator
    def validate(record)
      if record.make_current
        if record.before_any?
          record.errors.add(:started_at, "should
not be before any existing appointment")
        end
      end
    end
  end
end
  
```

a GtRubyClass (RoleAppointment)

```

create_table "role_appointments", :force => true,
  charset: "utf8mb3", collation: "utf8mb3_bin",
  force: :cascade do |t|
    t.integer "role_id"
    t.integer "person_id"
    t.datetime "created_at", precision: 6
    t.datetime "updated_at", precision: 6
    t.datetime "started_at", precision: 6
    t.datetime "ended_at", precision: 6
    t.string "content"
    t.integer "order"
    t.index ["ended_at"], name: "index_role_appointments_ended_at"
    t.index ["person_id"], name: "index_role_appointments_person_id"
    t.index ["role_id"], name: "index_role_appointments_role_id"
    t.index ["role_id", "person_id"], name: "index_role_appointments_role_id_person_id"
  end

create_table "role_translations", :force => true,
  charset: "utf8mb3", collation: "utf8mb3_bin",
  force: :cascade do |t|
    t.integer "role_id"
    t.string "locale"
    t.string "name"
    t.text "responsibilities"
    t.datetime "created_at", precision: 6
    t.datetime "updated_at", precision: 6
    t.index ["locale"], name: "index_role_translations_locale"
    t.index ["name"], name: "index_role_translations_name"
    t.index ["role_id"], name: "index_role_translations_role_id"
    t.index ["role_id", "locale"], name: "index_role_translations_role_id_locale"
  end

create_table "roles", :force => true,
  charset: "utf8mb3", collation: "utf8mb3_bin",
  force: :cascade do |t|
    t.datetime "created_at", precision: 6
    t.datetime "updated_at", precision: 6
    t.string "type", null: false
    t.boolean "permanent_secretary"
    t.boolean "cabinet_member"
    t.string "slug"
  end
  
```

Técnica post-libro

- Tras escribir Legacy Code First Aid Kit, ha experimentado con el moldable development.
- Única herramienta que conoce Glamorous Toolkit
- Ideal para sistemas viejos y críticos sin herramientas parecidas.
- Precio de entrada alto

Moldable development

- Herramientas personalizadas para cada problema a desarrollar.
- No solo queremos ver el código, queremos ver lo que el código **significa** en el contexto de nuestro sistema.
- Solo existe Glamorous Toolkit hasta la fecha.



```
curl -o feenk-releaser-LsS https://github.com/feenkcom/releaser-rs/releases/download/v0.14.0/feenk-rs-LsS
```

```
chmod +x feenk-releaser
```

```
to let the access to the image."
    self class CommandLinePasswordManager hasPasswordSet
    ifTrue: [ Smalltalk snapshot: false andQuit: true ].
    self handleExit: exit ] ] in
PharoCommandLineHandler(BasicCommandLineHandler)>>activate in Block:
[ [ self handleArgument: self firstArgument ]...
[self value.

"IMPORTANT: Do not step over next line of code. See
method comments for details"
Processor terminateRealActive] in
FullBlockClosure(BlockClosure)>>newProcess in Block: [self value....
[0m[97;41;22mError:[91;49;1m Command cd "/Users/tudor/jenkins/
workspace/feenkcom_gtoolkit_main/gt-releaser" && "/Users/tudor/
jenkins/workspace/feenkcom_gtoolkit_main/gt-releaser/
GlamorousToolkit.app/Contents/MacOS/GlamorousToolkit-cli" "/Users/
tudor/jenkins/workspace/feenkcom_gtoolkit_main/gt-releaser/
GlamorousToolkit.image" "releasegtoolkit" "--strategy=patch" "--
expected=1.0.1571" "--verbose" failed. See install.log or install-
errors.log for more info[0m
[37;49;2m[0m
script returned exit code 1
```

```
Caught exception: hudson.AbortException: script returned exit code 1; currentBuild.result: null
```

```
Send Slack Message
```


Conclusion



Hi, I'm Nicolas 🍌

I write about VS Code, web development and life in general.

[Articles](#)

[Talks](#)

[Side projects](#)

[Who I am](#)

[My recommendations](#)

[🕒 understandlegacycode.com](#)

[🌸 refactoringjavascript.dev](#)



[Switch to dark theme](#)

[FR](#) [Basculer en français](#)

Who am I?

My name is **Nicolas Carlo**.

I am a freelance web developer living and working from Montreal, Canada.

I help people build maintainable softwares. I'm specialized in working with Legacy Code.

I'm really into community events 🍌 I organize the **Software Crafters** and the **React** Montréal guilds. I also give a hand to other local communities, such as the **TypeScript** and **Elixir** ones. In the past, I've also organized **The Legacy of SoCraTes**, **SoCraTes Canada**, and **MenderCon** conferences.

Before

In Paris, I organized:

- the meetup **Backbone.js Paris**
- the meetup **Paris WebComponents**
- the annual conference **Best of Web Paris**.

From 2016 to 2018, I was Consultant at **OCTO Technology**. I supported clients running their Agile projects, I did web development and I coached about Software Craftsmanship practices.

Rescue your Legacy codebase *quickly and safely.* 🧑‍🔧

Learn how to refactor an existing codebase as you go using specific techniques to incrementally make it easier to maintain.

Sign up for my newsletter to get a **free chapter preview** on performing incremental refactorings so you can stop and ship at any time.

