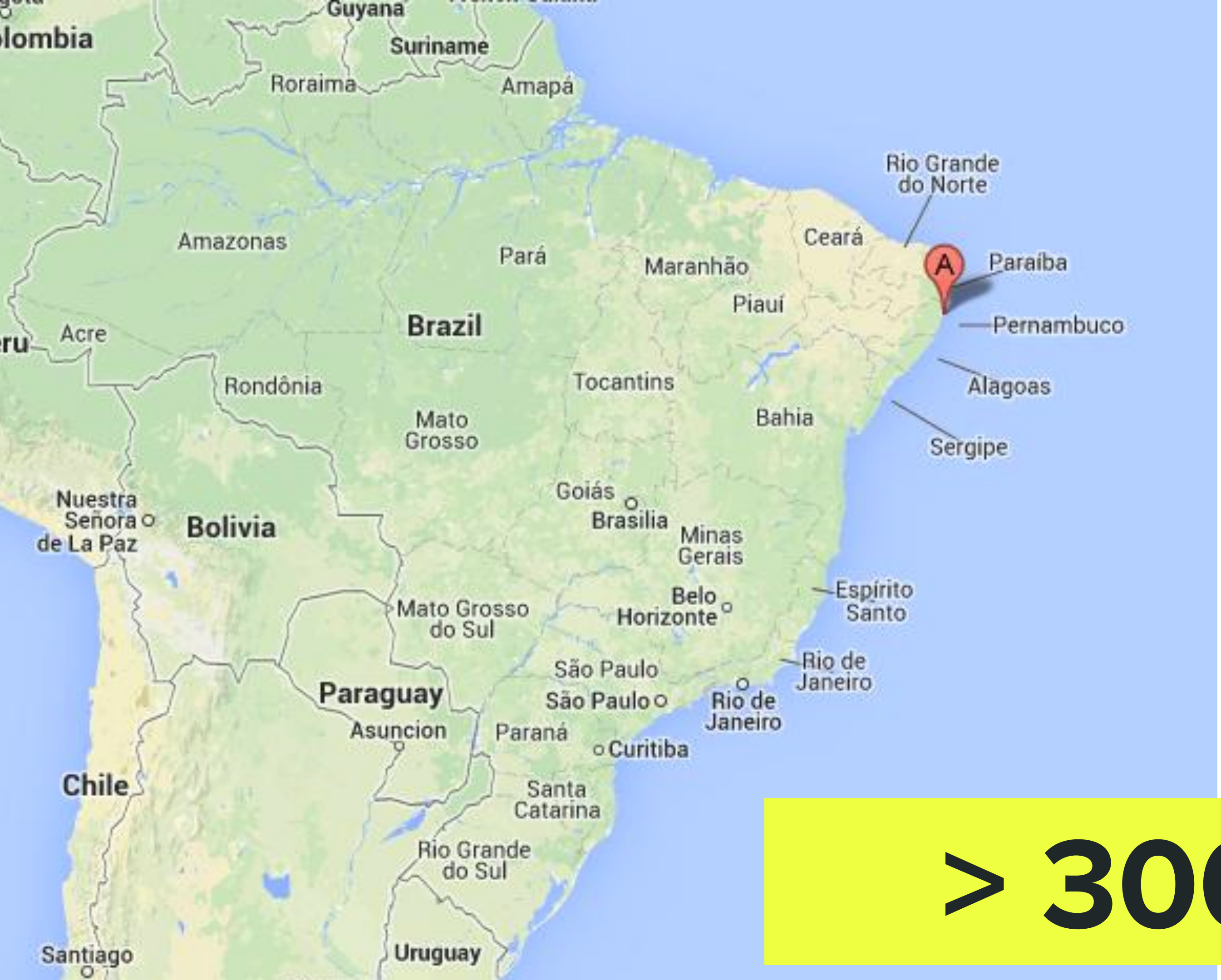
De Produtos a Linhas de Produtos: Um breve panorama de teorias formais para análise e evolução de Linhas de Produtos de Software

Leopoldo Teixeira
(lmt@cin.ufpe.br)

Centro de Informática UFPE

Universidade Federal de Pernambuco

Unterstützt von / Supported by Alexander von Humboldt Stiftung/Foundation



Centro de Informática UFPE

Universidade Federal de Pernambuco

> 3000km

Centro de Informática

UFPE

Pós-graduação Acadêmica em Ciência da Computação é nota máxima pela CAPES

Apenas oito instituições no país alcançaram a nota 7

https://portal.cin.ufpe.br/

@cinufpe

Centro de Informática UFPE

Centro de Informática UFPE

@CInUFPE

50 anos

Centro de Informática UFPE

CBSOFT'25

XVI CONGRESSO BRASILEIRO DE SOFTWARE: TEORIA E PRÁTICA

22 A 26 DE SETEMBRO | RECIFE/PE

cin.ufpe.br

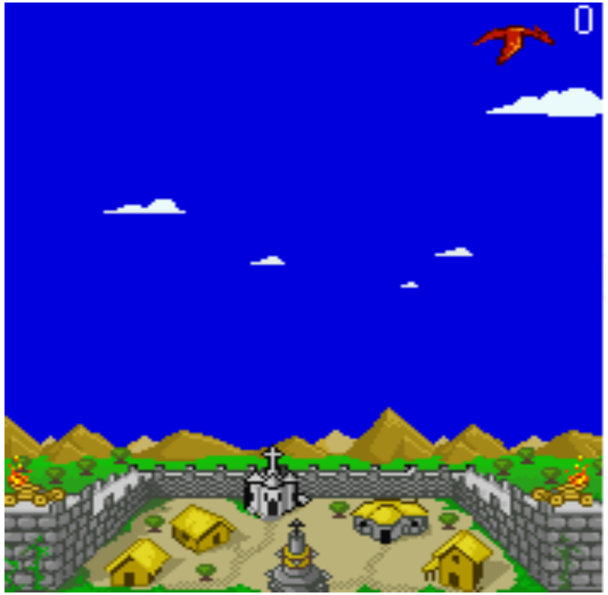
voltando ao tema principal...

Centro de Informática UFPE

Universidade Federal de Pernambuco

De um produto (Rain of Fire, circa 2005...)

Nuvens Movendo

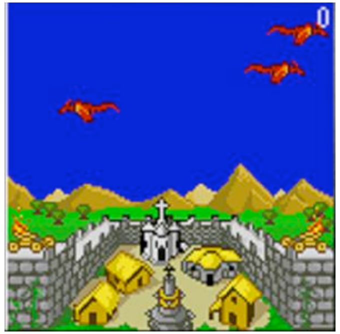


meantime mobile creations

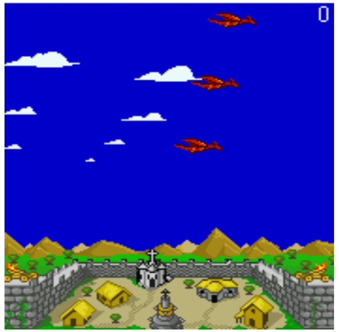
A produtos (reuso oportunista, ad hoc...)

produtos similares

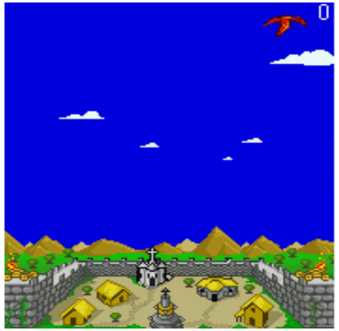
Sem Nuvens



Nuvens Estáticas



Nuvens Movendo



meantime mobile creations

Centro de Informática UFPE

Universidade Federal de Pernambuco

...a Linhas de Produtos de Software

artefatos reusáveis

.java

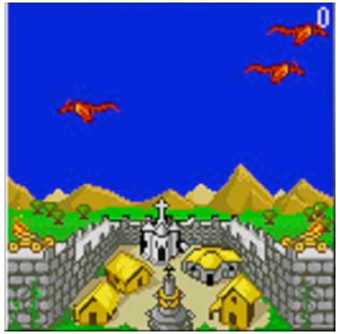
.aj

.xml

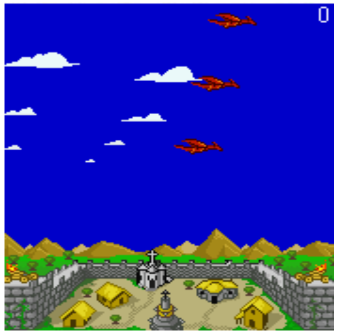
.jpg

produtos similares

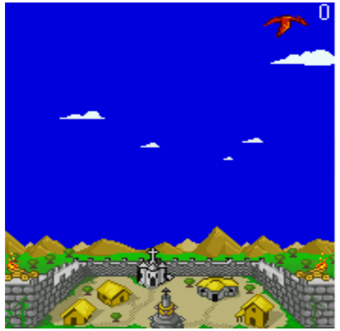
Sem Nuvens



Nuvens Estáticas

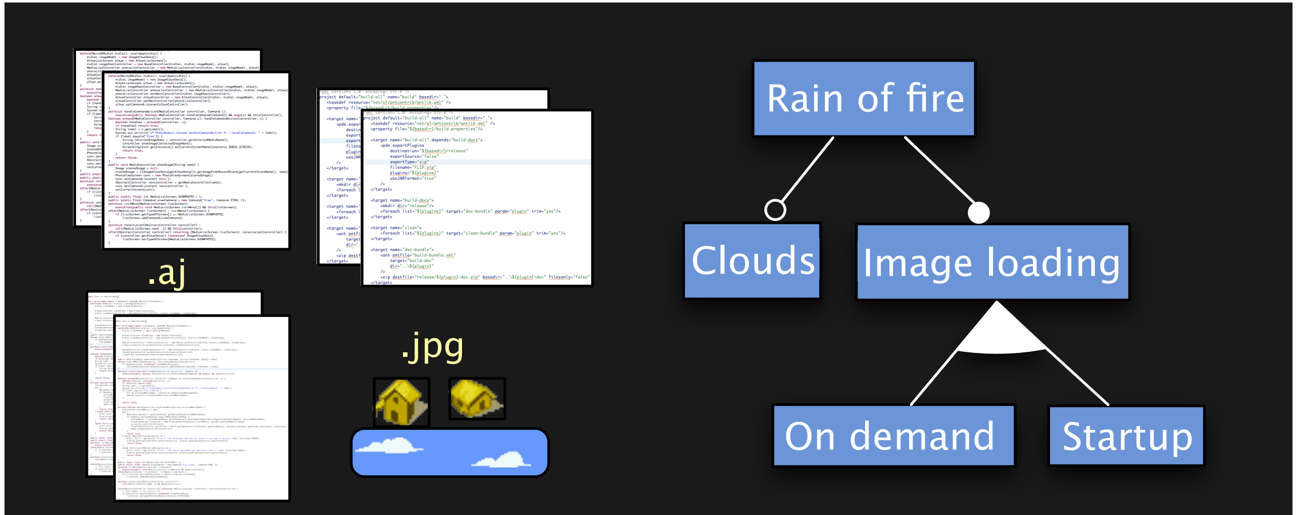


Nuvens Movendo



meantime mobile creations

Artefatos vão além de código!



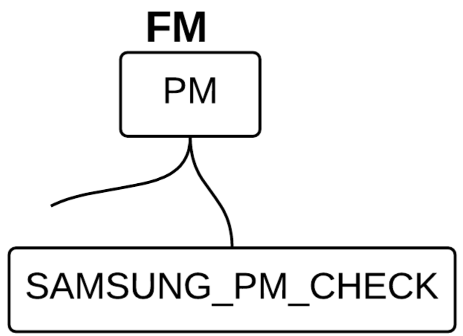
Rain of fire	Rain.java, Common-Clouds.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj
Clouds	Clouds.java

```
#endif
        int on_rq;

        int prio, static_prio, normal_prio;
        unsigned int rt_priority;
        const struct sched_class *sched_class;
        struct sched_entity se;
        struct sched_rt_entity rt;
#ifdef CONFIG_CGROUP_SCHED
        struct task_group *sched_task_group;
#endif

#ifdef CONFIG_PREEMPT_NOTIFIERS
        /* list of struct preempt_notifier: */
        struct hlist_head preempt_notifiers;
#endif

#ifdef CONFIG_BLK_DEV_IO_TRACE
        unsigned int btrace_seq;
#endif
        ...
```

e
ca

UNIVERSIDADE FEDERAL DE PERNAMBUCO

FM



Kconfig

```
config SAMSUNG_PM_CHECK
bool "S3C2410 PM Suspend
Memory CRC"
depends on PM
select CRC32
...
```

Configuration Knowledge

CONFIG_SAMSUNG_PM_CHECK	pm-check.o
...	...

Makefile

```
obj-$(CONFIG_SAMSUNG_
PM_CHECK) += pm-check.o
...
```

Asset Mapping

$$\left\{ \begin{array}{c} \text{pm-check.c} \mapsto \\ \dots \end{array} \right\}$$

Implementation

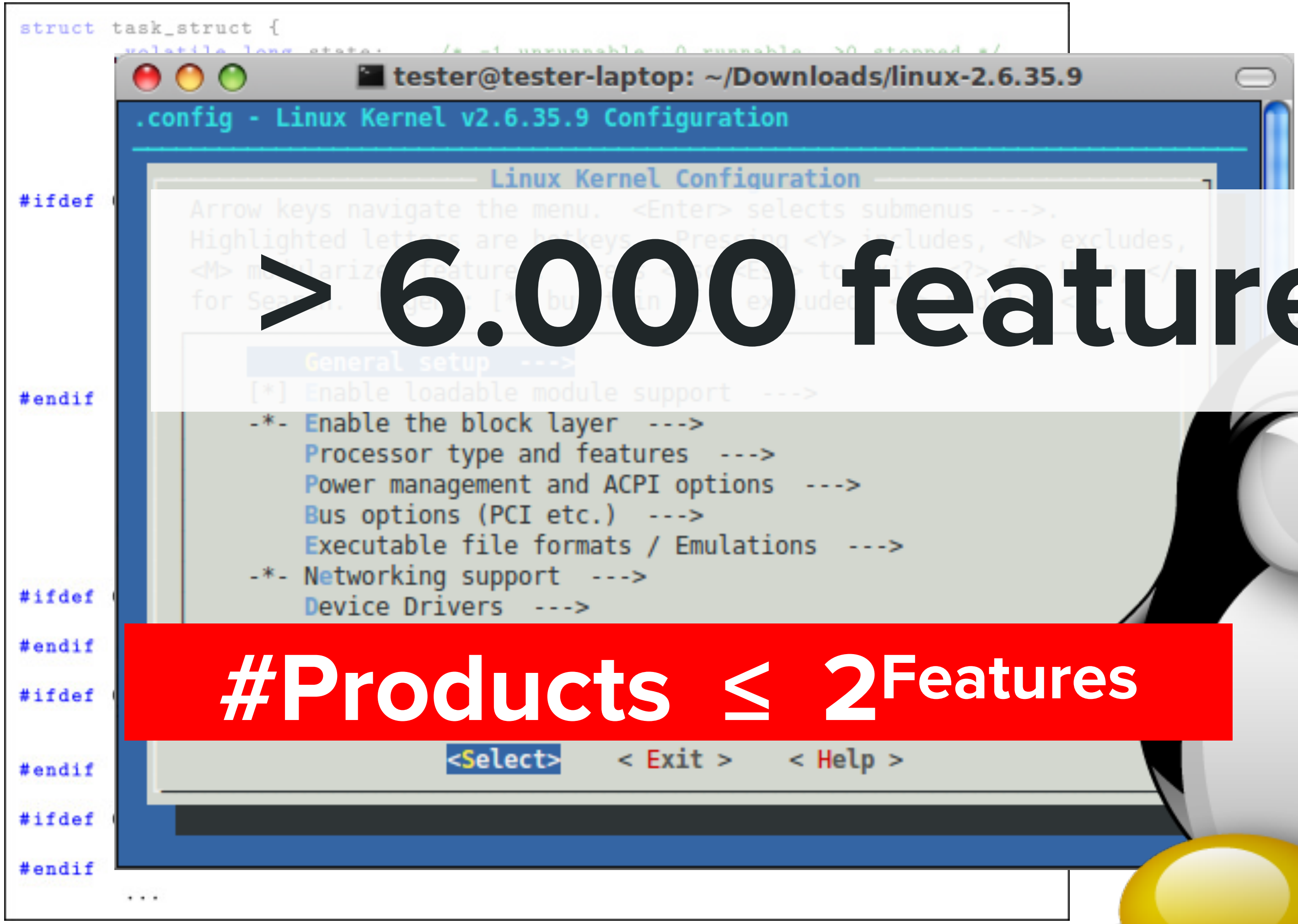
```
#ifdef
    CONFIG_SAMSUNG_PM_CHECK
    ...
```

pm-check.c

.ufpe.br

Centro de Informática UFPE

Universidade Federal de Pernambuco



```
struct task_struct {
    volatile long state;    /* -1 unrunnable, 0 runnable, >0 stopped */

#ifdef

#endif

#ifdef
#endif
#ifdef

#endif
#ifdef
#endif
    ...
```

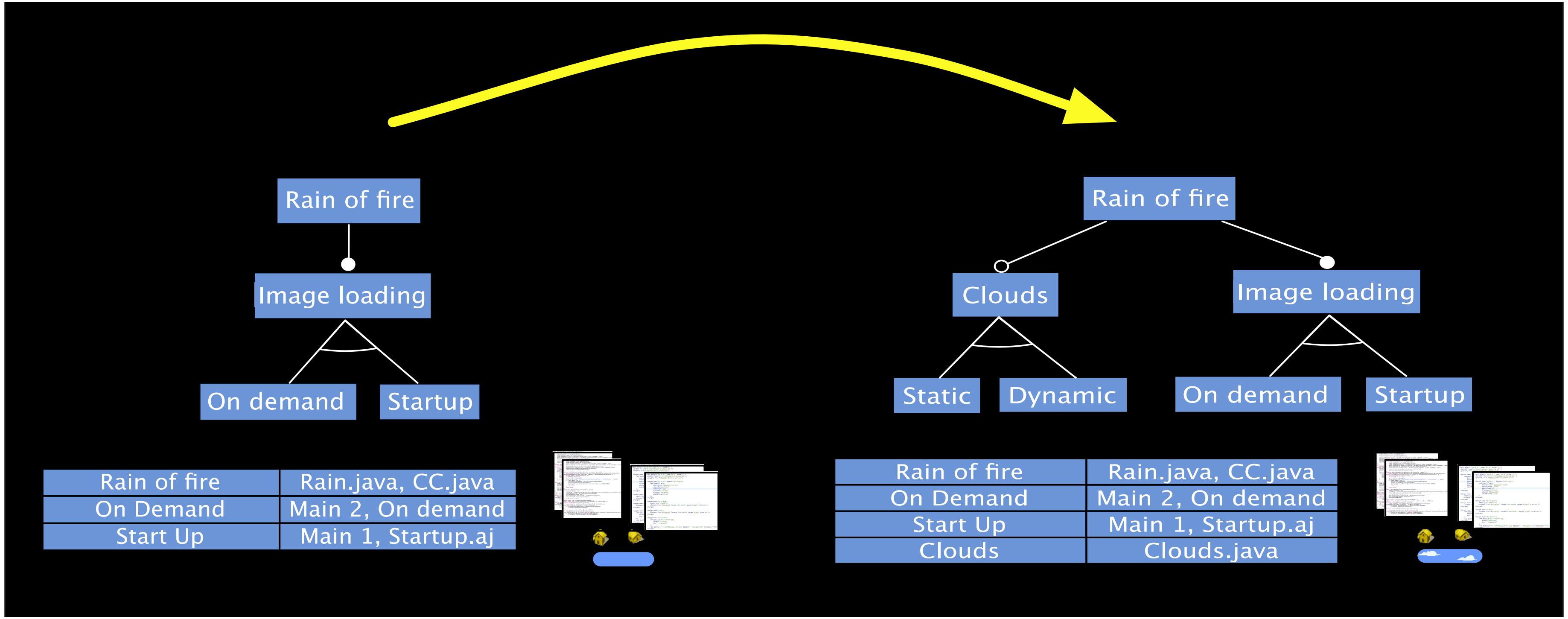
```
tester@tester-laptop: ~/Downloads/linux-2.6.35.9
.config - Linux Kernel v2.6.35.9 Configuration
                      Linux Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularize features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

        General setup  --->
    [*] Enable loadable module support  --->
    -*- Enable the block layer  --->
        Processor type and features  --->
        Power management and ACPI options  --->
        Bus options (PCI etc.)  --->
        Executable file formats / Emulations  --->
    -*- Networking support  --->
        Device Drivers  --->

              <Select>    < Exit >    < Help >
```

> 6.000 features

#Products $\leq 2^{\text{Features}}$

Sistemas ricos em variabilidade <u>evoluem</u> de múltiplas maneiras...

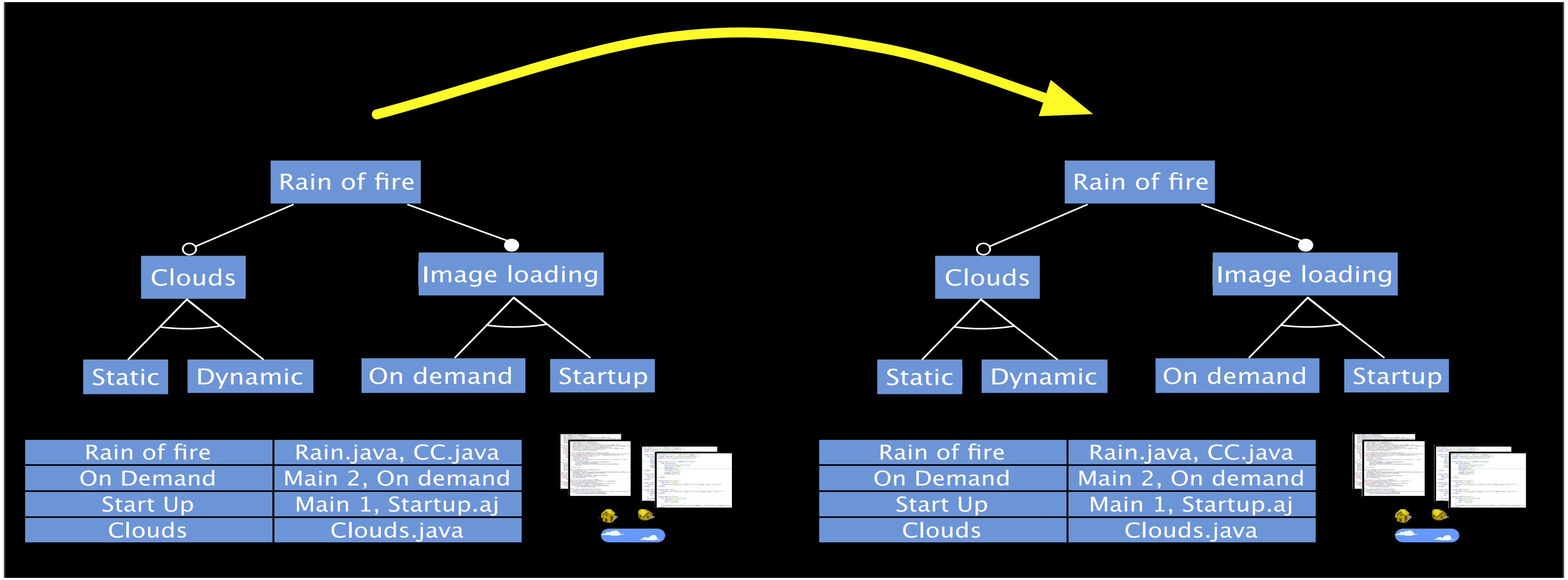
Adicionando funcionalidade



Rain of fire	Rain.java, CC.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj

Rain of fire	Rain.java, CC.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj
Clouds	Clouds.java

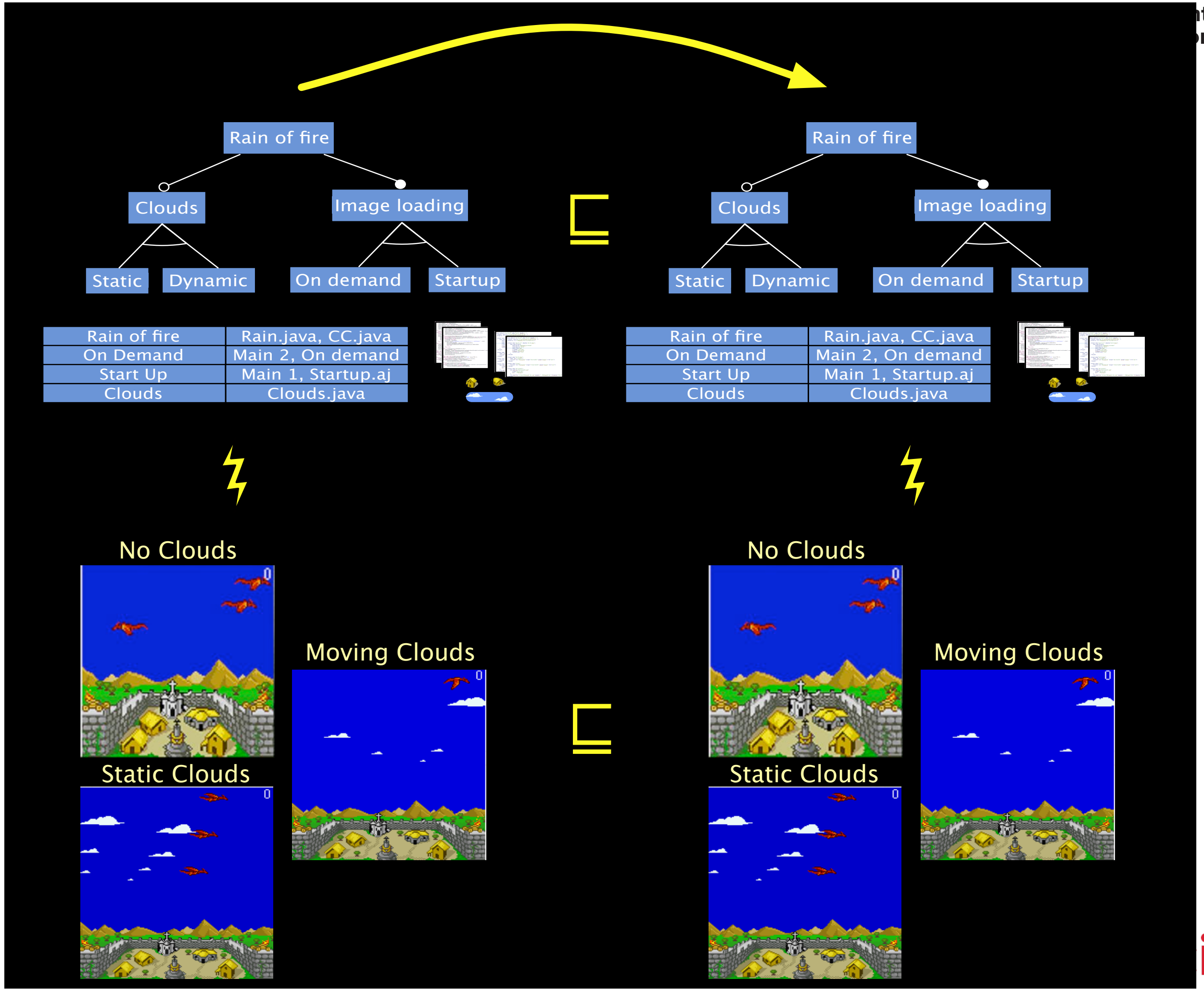
Refatorando artefatos existentes



Rain of fire	Rain.java, CC.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj
Clouds	Clouds.java

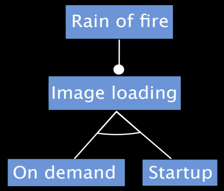
Rain of fire	Rain.java, CC.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj
Clouds	Clouds.java

...como podemos apoiar a evolução, garantindo que foi realizada de forma <u>segura</u>?

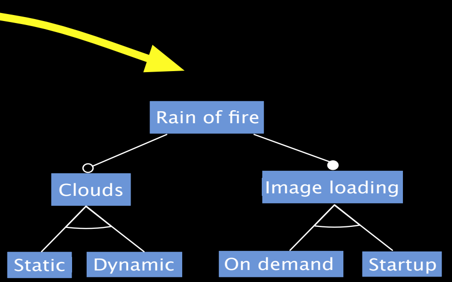


Rain of fire	Rain.java, CC.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj
Clouds	Clouds.java

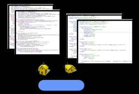
Rain of fire	Rain.java, CC.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj
Clouds	Clouds.java



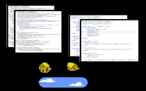
$\sqsubseteq$



Rain of fire	Rain.java, CC.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj



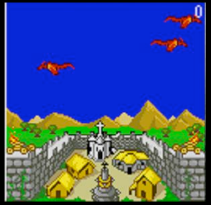
Rain of fire	Rain.java, CC.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj
Clouds	Clouds.java



ϟ

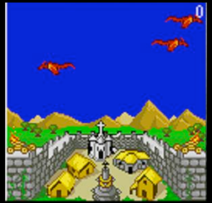
ϟ

No Clouds

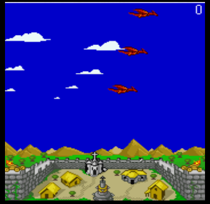


$\sqsubseteq$

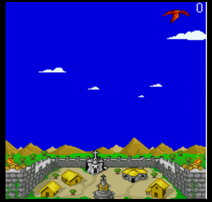
No Clouds



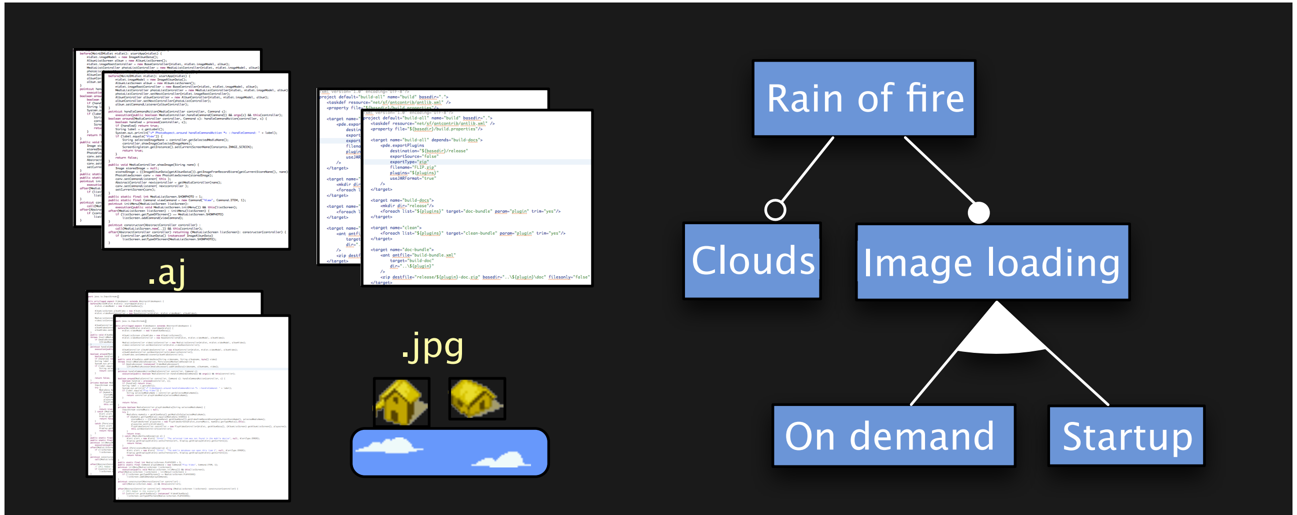
Static Clouds



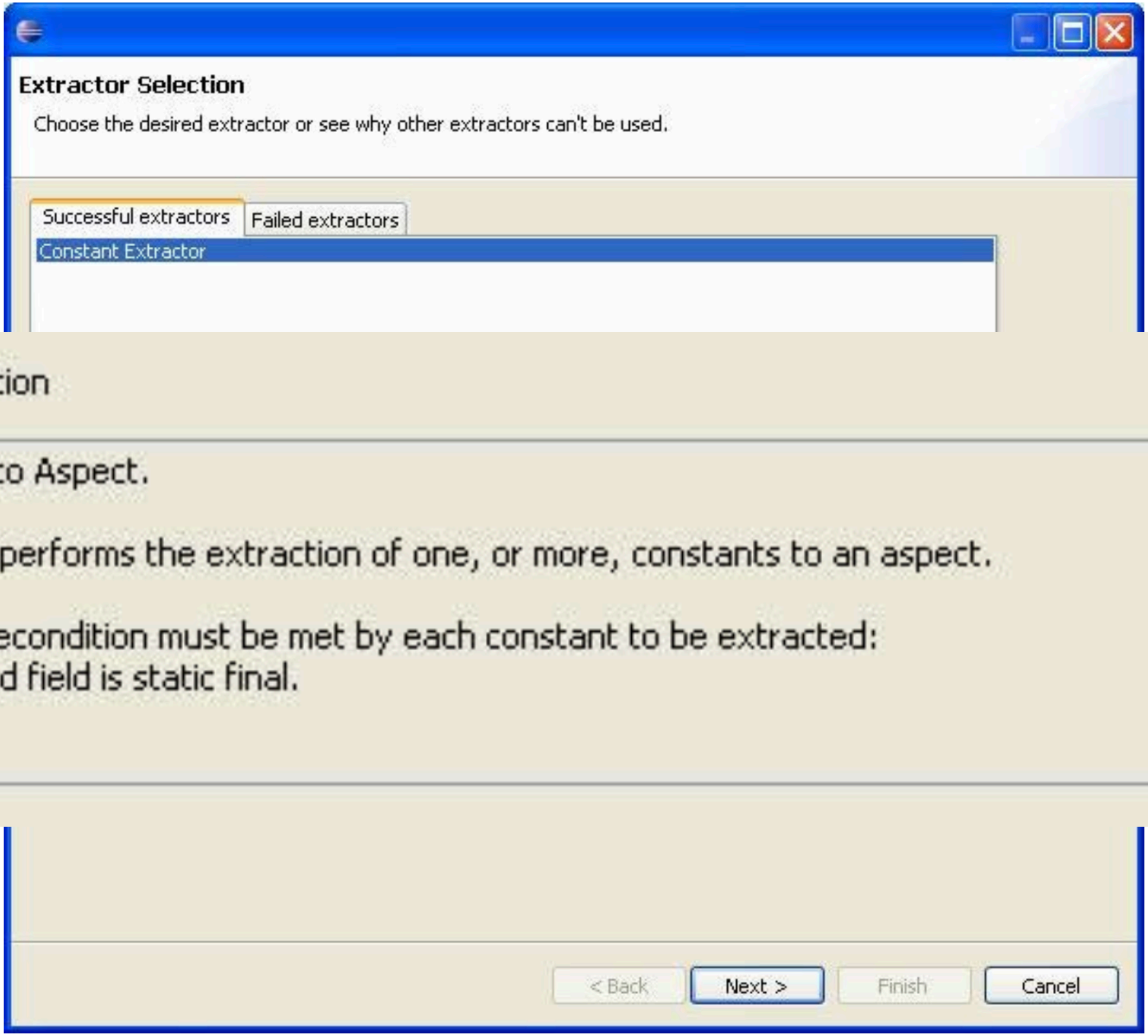
Moving Clouds



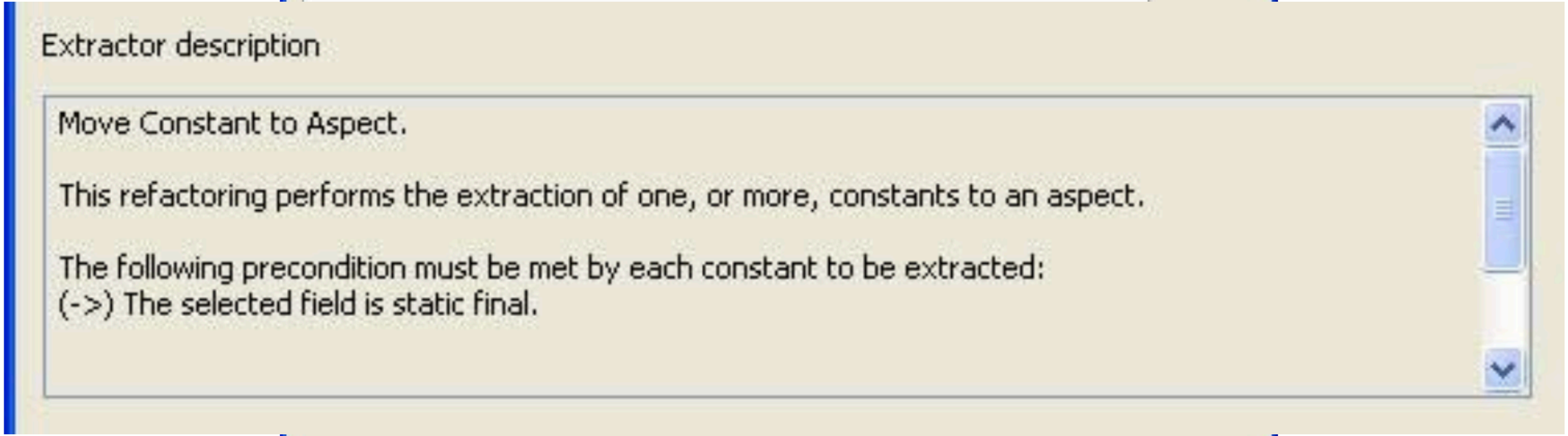
Considerando também os múltiplos artefatos...



Rain of fire	Rain.java, Common-Clouds.java
On Demand	Main 2, On demand
Start Up	Main 1, Startup.aj
Clouds	Clouds.java


Extractor Selection

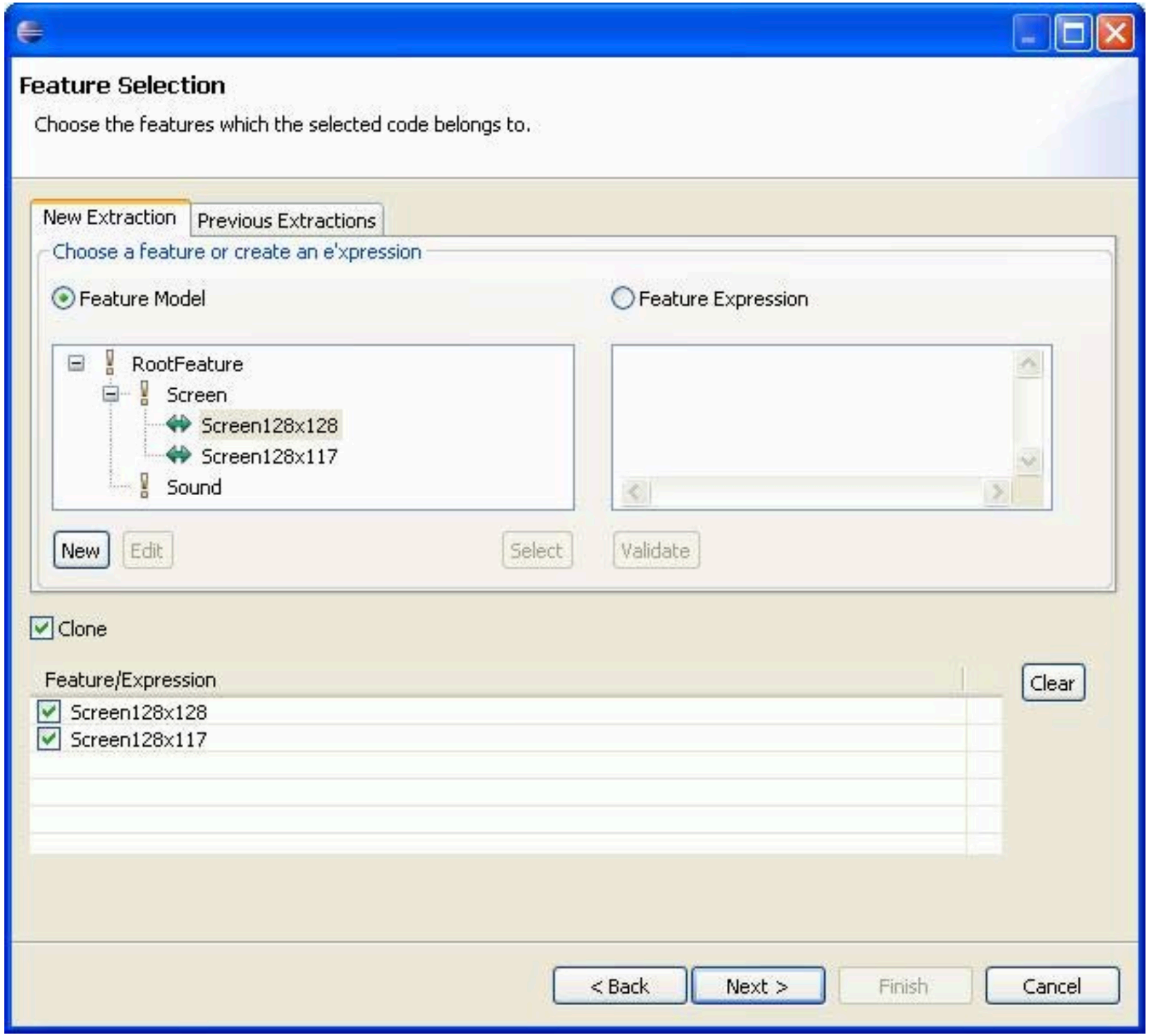
Choose the desired extractor or see why other extractors can't be used.


Extractor description

Move Constant to Aspect.

This refactoring performs the extraction of one, or more, constants to an aspect.

The following precondition must be met by each constant to be extracted:
(->) The selected field is static final.



Feature Selection

Choose the features which the selected code belongs to.

New Extraction | Previous Extractions

Choose a feature or create an e'xpression

[✓ Feature Model] Feature Expression

- RootFeature
  - Screen
    - Screen128x128
    - Screen128x117
  - Sound

New Edit Select Validate

[✓] Clone

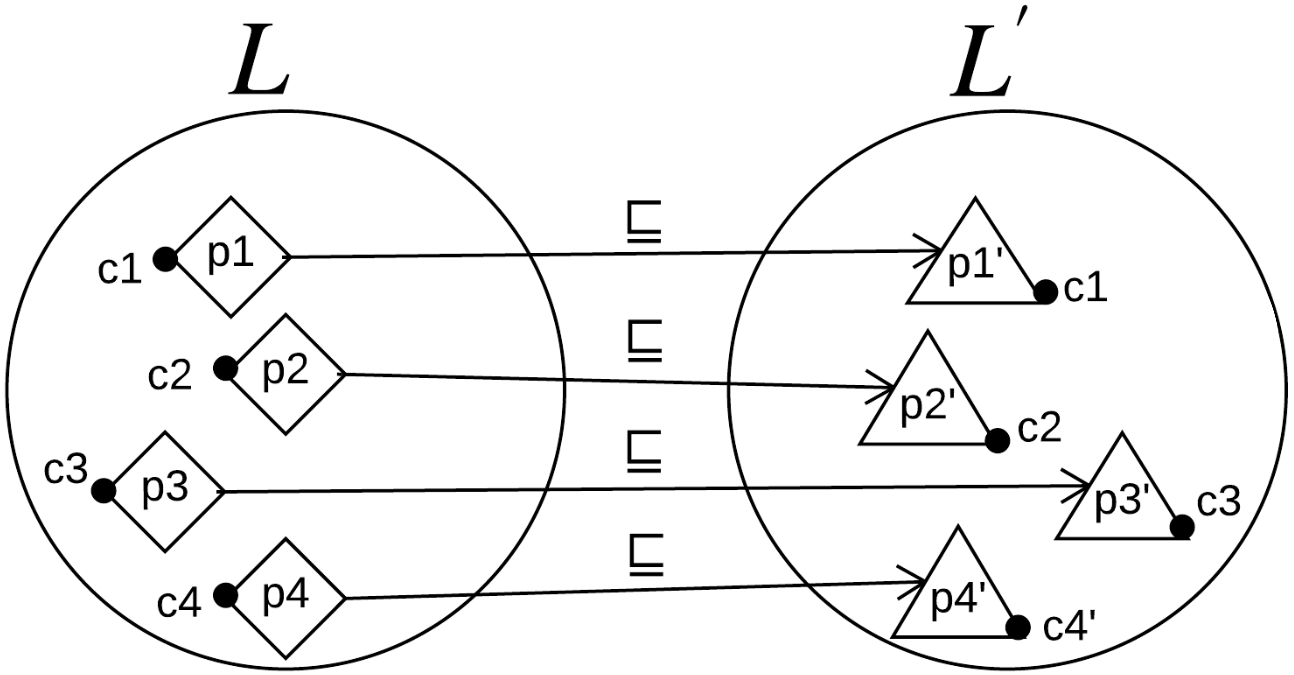
Feature/Expression
✓ Screen128x128
✓ Screen128x117

Clear

< Back Next > Finish Cancel

Ideia base: como garantir evolução <u>segura</u> dos produtos?

Evolução segura, foco nos produtos!



Todo produto de L tem um produto compatível em L'

[ICTAC'10, TCS'12]

Formalização: Refinamento de LPS

$$L \sqsubseteq L'$$

quando

$$\forall\, p \in [[L]] \cdot \exists\, p' \in [[L']] \cdot p \sqsubseteq p'$$

[ICTAC'10, TCS'12, SPLC'15]

Linhas de Produtos

Produtos

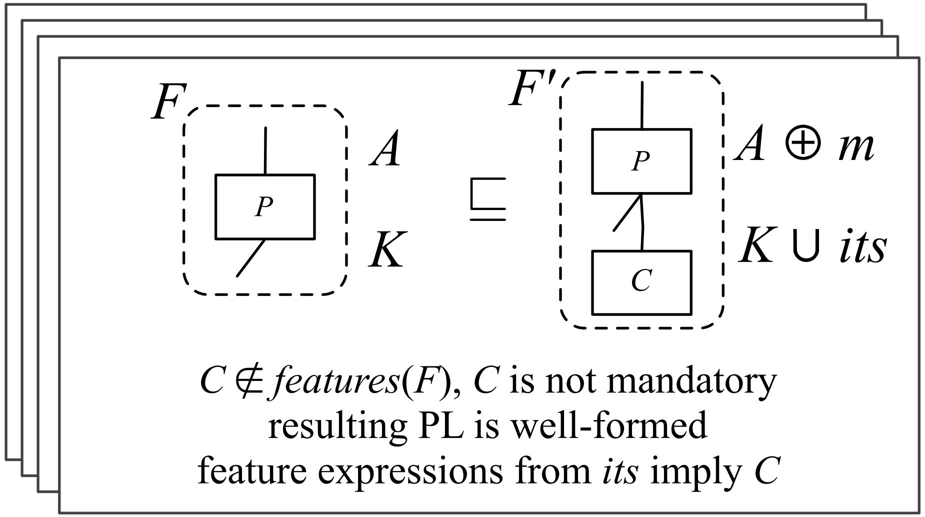
$$\begin{tikzcd} \boxed{L \sqsubseteq L'} & \boxed{p \sqsubseteq p'} \arrow[l] \end{tikzcd}$$

quando

$$\forall\, p \in [[L]] \cdot \exists\, p' \in [[L']] \cdot p \sqsubseteq p'$$

[ICTAC’10, TCS’12, SPLC’15]

Principais aplicações da teoria

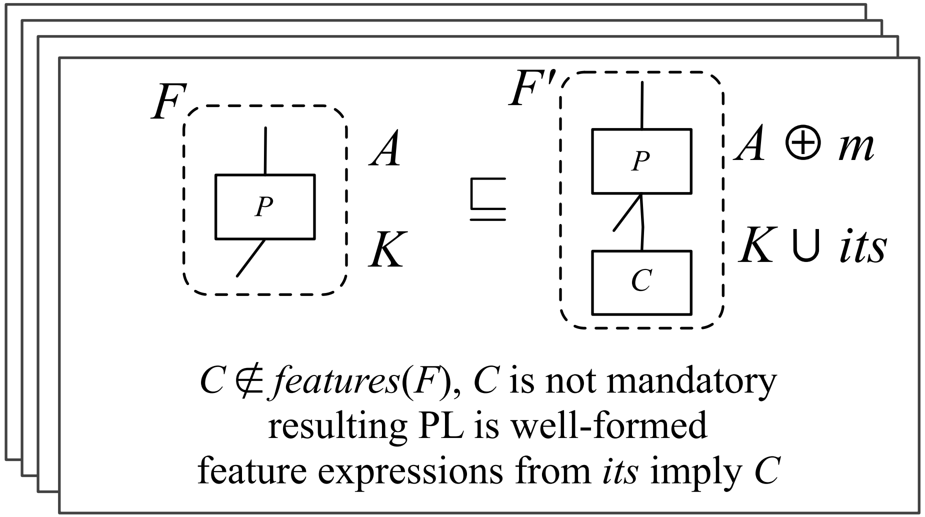


$C \notin features(F)$, C is not mandatory
resulting PL is well-formed
feature expressions from *its* imply C

Templates/Padrões (*a priori*)
foco principal de hoje

Centro de Informática UFPE · Universidade Federal de Pernambuco

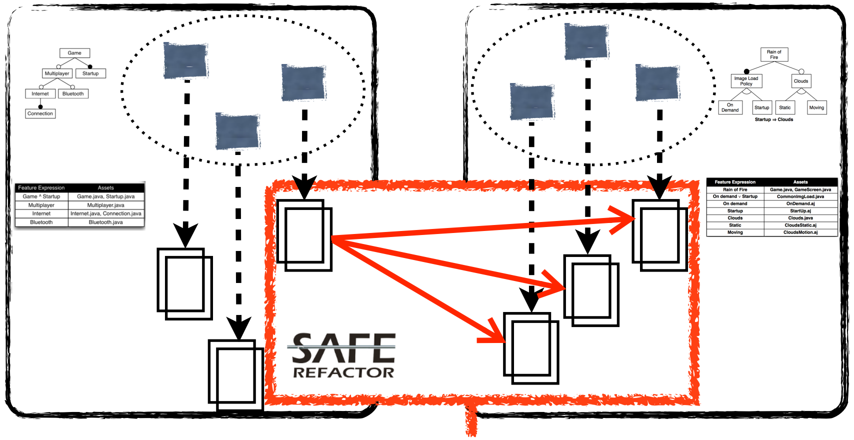
Principais aplicações da teoria



$C \notin features(F)$, C is not mandatory
resulting PL is well-formed
feature expressions from *its* imply C

Templates/Padrões (*a priori*)
foco principal de hoje

Verificação (*a posteriori*)

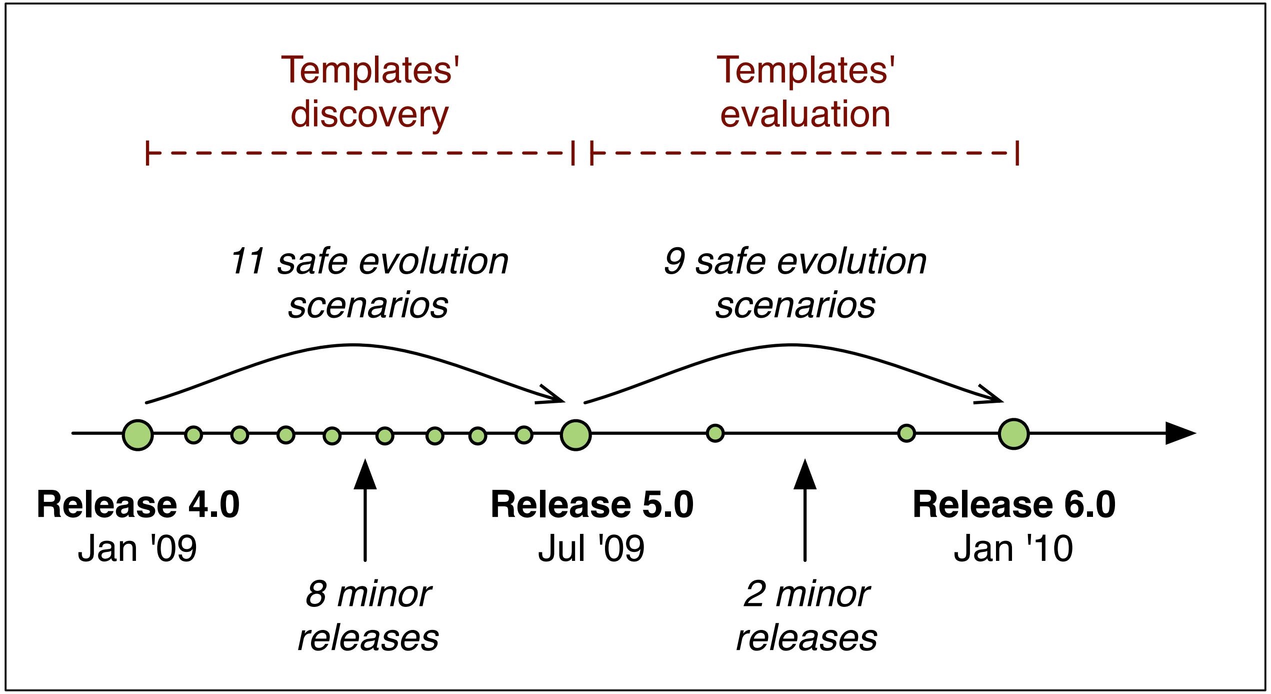


Feature Expression	Assets
Game ^ Startup	Game.java, Startup.java
Multiplayer	Multiplayer.java
Internet	Internet.java, Connection.java
Bluetooth	Bluetooth.java

Feature Expression	Assets
Rain of Fire	Game.java, GameScreen.java
On demand ∨ Startup	CommonImgLoad.java
On demand	OnDemand.aj
Startup	StartUp.aj
Clouds	Clouds.java
Static	CloudsStatic.aj
Moving	CloudsMotion.aj

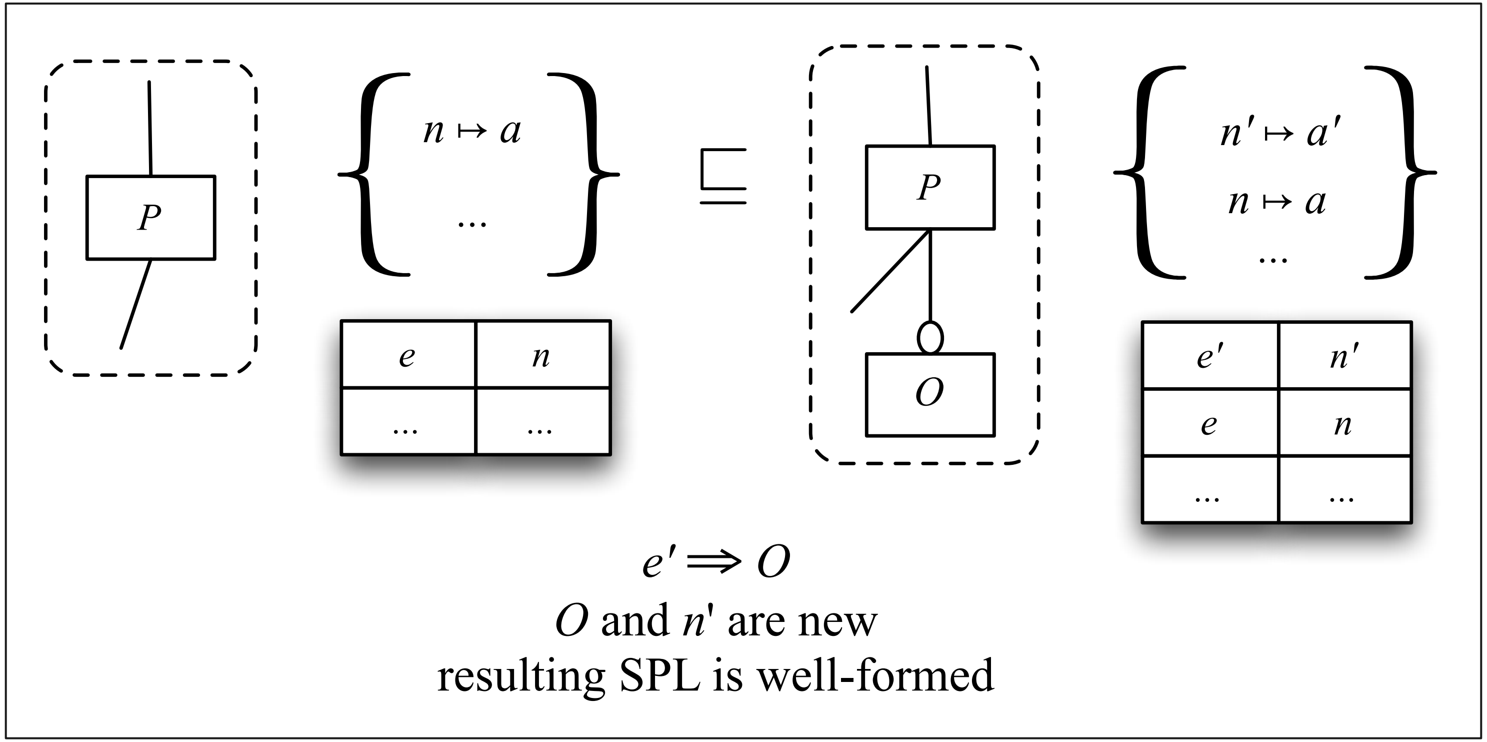
Como definir estes padrões recorrentes de evolução?

Minerando mudanças recorrentes (padrões/templates)



[GPCE'11, SPLC'15, JSS'15] e também relacionados: [SPLC'13, EMSE'16, SBCARS'16]

Estabelecendo padrões de evolução (templates)



e	n
...	...

e'	n'
e	n
...	...

$e' \Rightarrow O$

O and n' are new

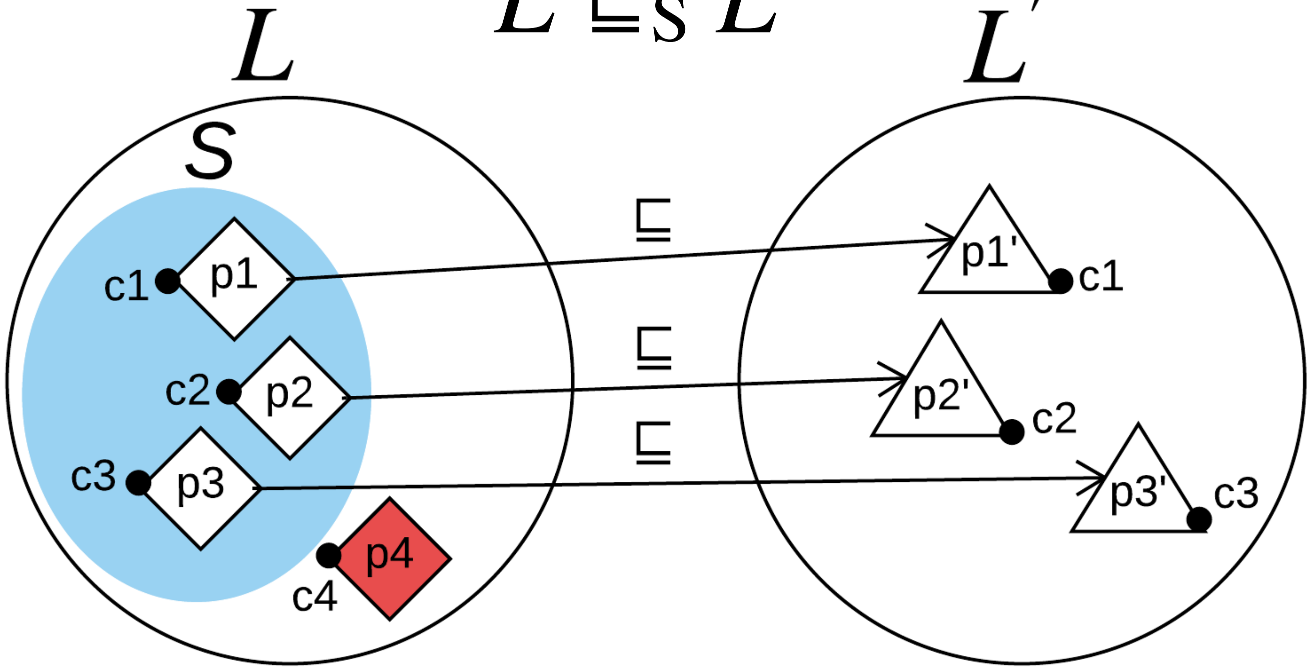
resulting SPL is well-formed

[GPCE'11, SPLC'15, JSS'15] e também relacionados: [SPLC'13, EMSE'16, SBCARS'16]

…mas, e se quisermos remover uma funcionalidade?

Evolução parcialmente segura

$$L \sqsubseteq_S L'$$

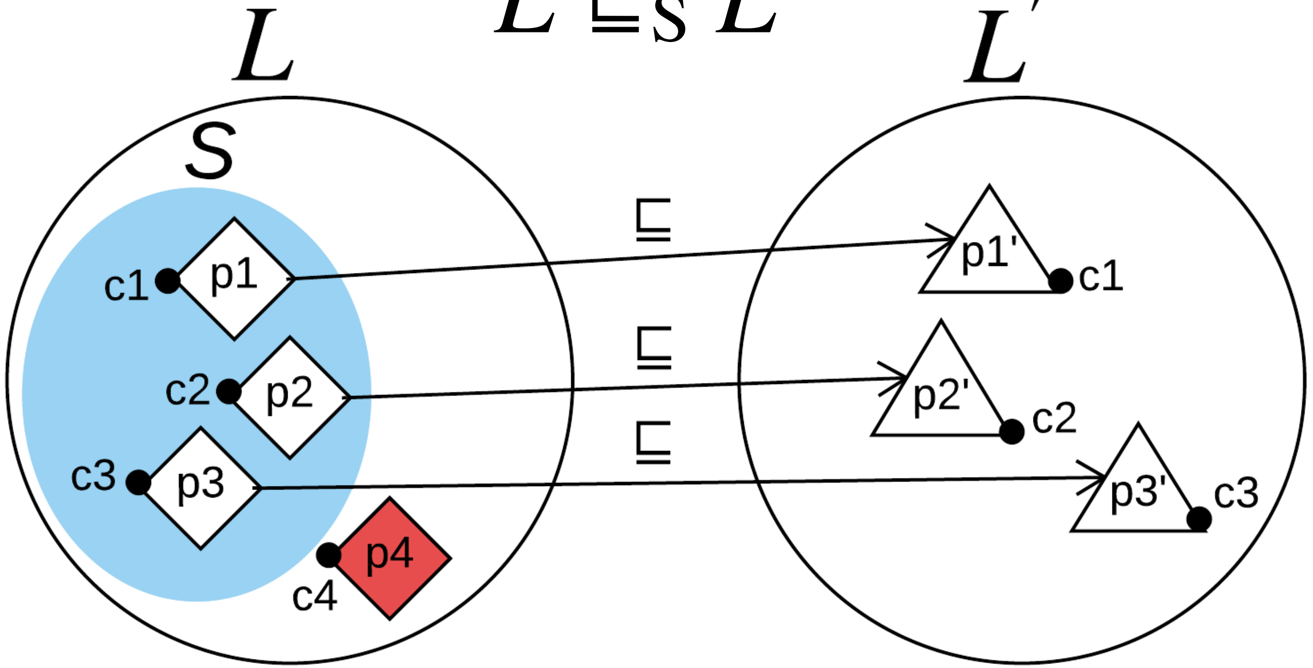


Não há produto compatível com p4 em L'

ideia chave: análise de impacto

A garantia é apenas para produtos dentro de S

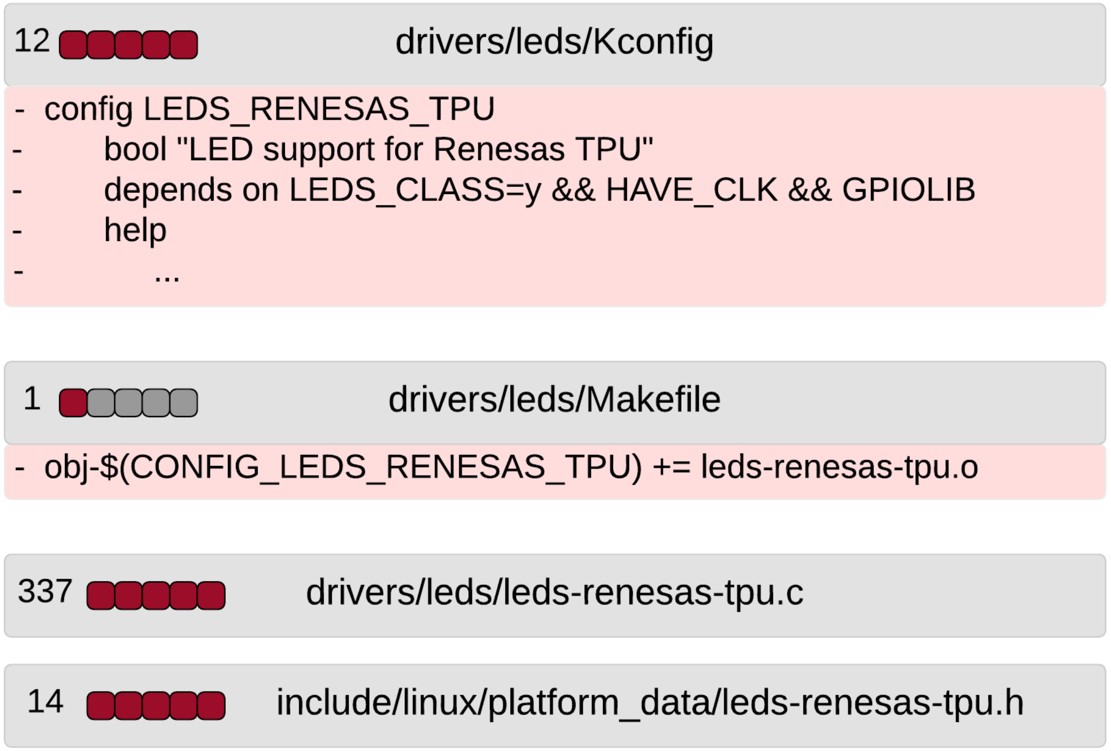
$$L \sqsubseteq_S L'$$



Não há produto compatível com p4 em L'

[SPLC'16, JSS'19]

Também identificamos operações recorrentes...



12 🟥🟥🟥🟥🟥 drivers/leds/Kconfig

```
-  config LEDS_RENESAS_TPU
-        bool "LED support for Renesas TPU"
-        depends on LEDS_CLASS=y && HAVE_CLK && GPIOLIB
-        help
-              ...
```

1 🟥⬜⬜⬜⬜ drivers/leds/Makefile

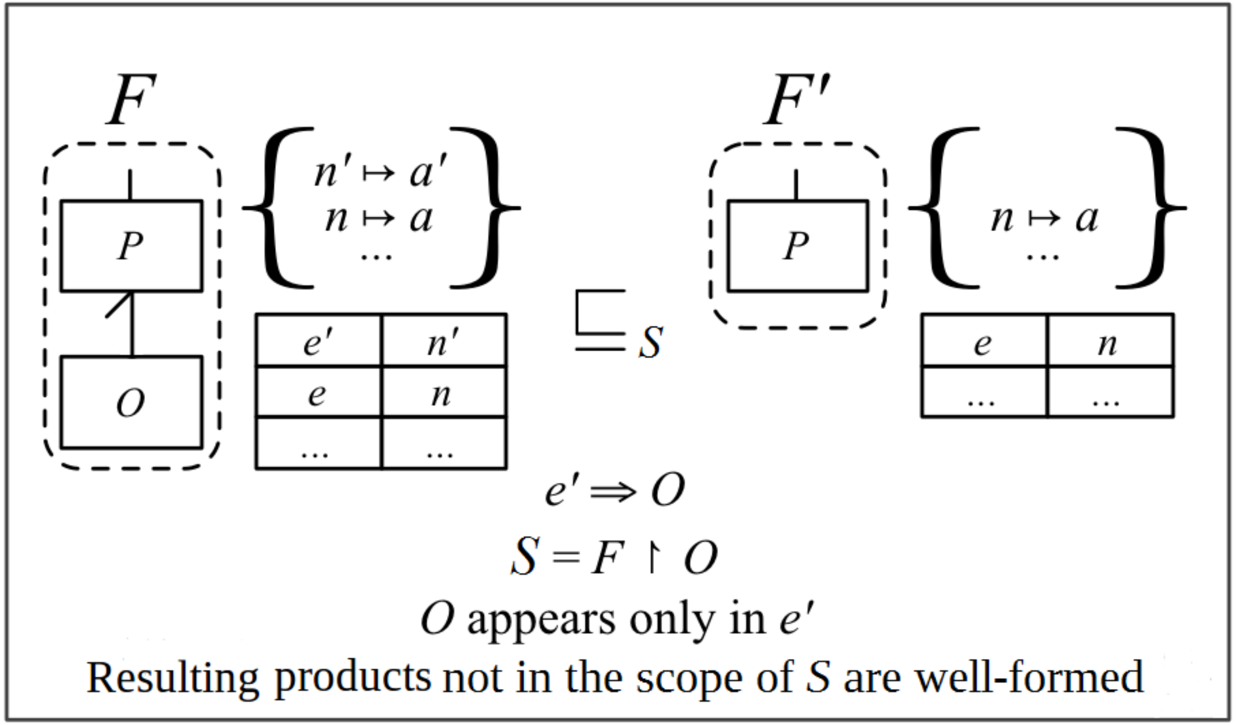
```
-  obj-$(CONFIG_LEDS_RENESAS_TPU) += leds-renesas-tpu.o
```

337 🟥🟥🟥🟥🟥 drivers/leds/leds-renesas-tpu.c

14 🟥🟥🟥🟥🟥 include/linux/platform_data/leds-renesas-tpu.h

Commit ae3e4c2776 from the Linux kernel

Definindo novos padrões/templates...



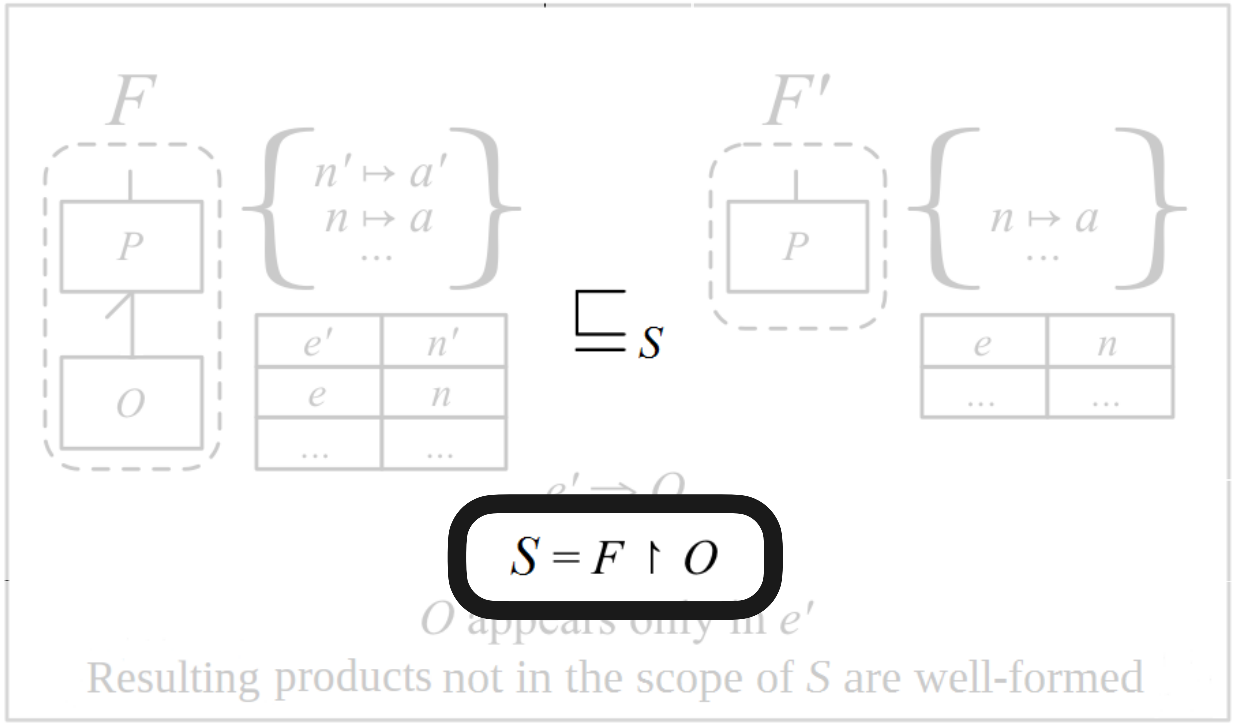
$$e' \Rightarrow O$$

$$S = F \upharpoonright O$$

O appears only in e'

Resulting products not in the scope of S are well-formed

Remover features



$$S = F \restriction O$$

S é o conjunto de configurações geradas a partir de F que não tem a *feature* O

Estudo empírico sobre expressividade de templates

Linux

15.373 KLOC
43036 Stars
16.323 Contributors
67310 Commits analysed
2 Sep 2013 - 3 Aug 2014
Versions 3.11 - 3.16

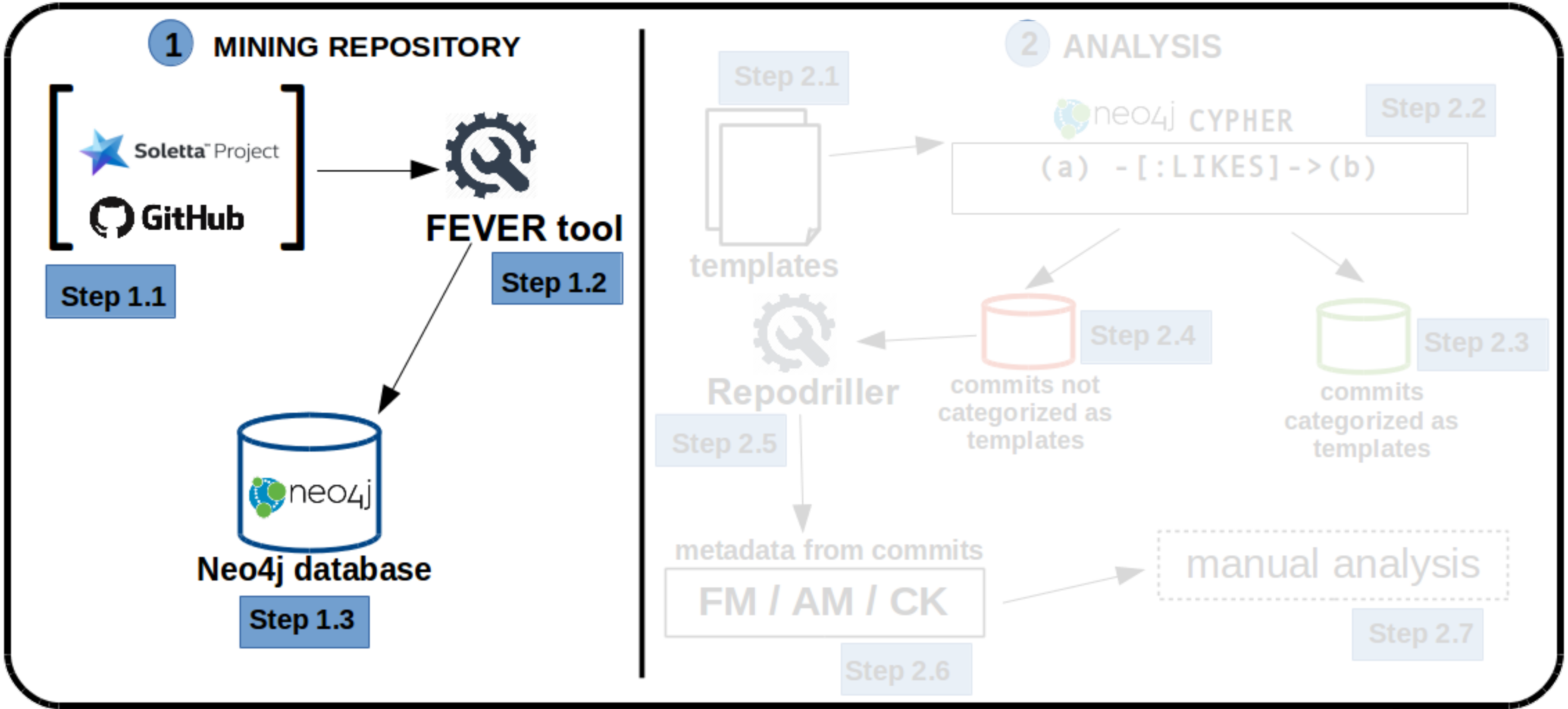
Soletta Project

170 KLOC
149 Stars
47 Contributors
2300 Commits analysed
26 Jun 2015 - 9 Apr 2016
Versions v1_beta0 - v1_beta18

[VaMoS’19, Karine’s MSc dissertation]

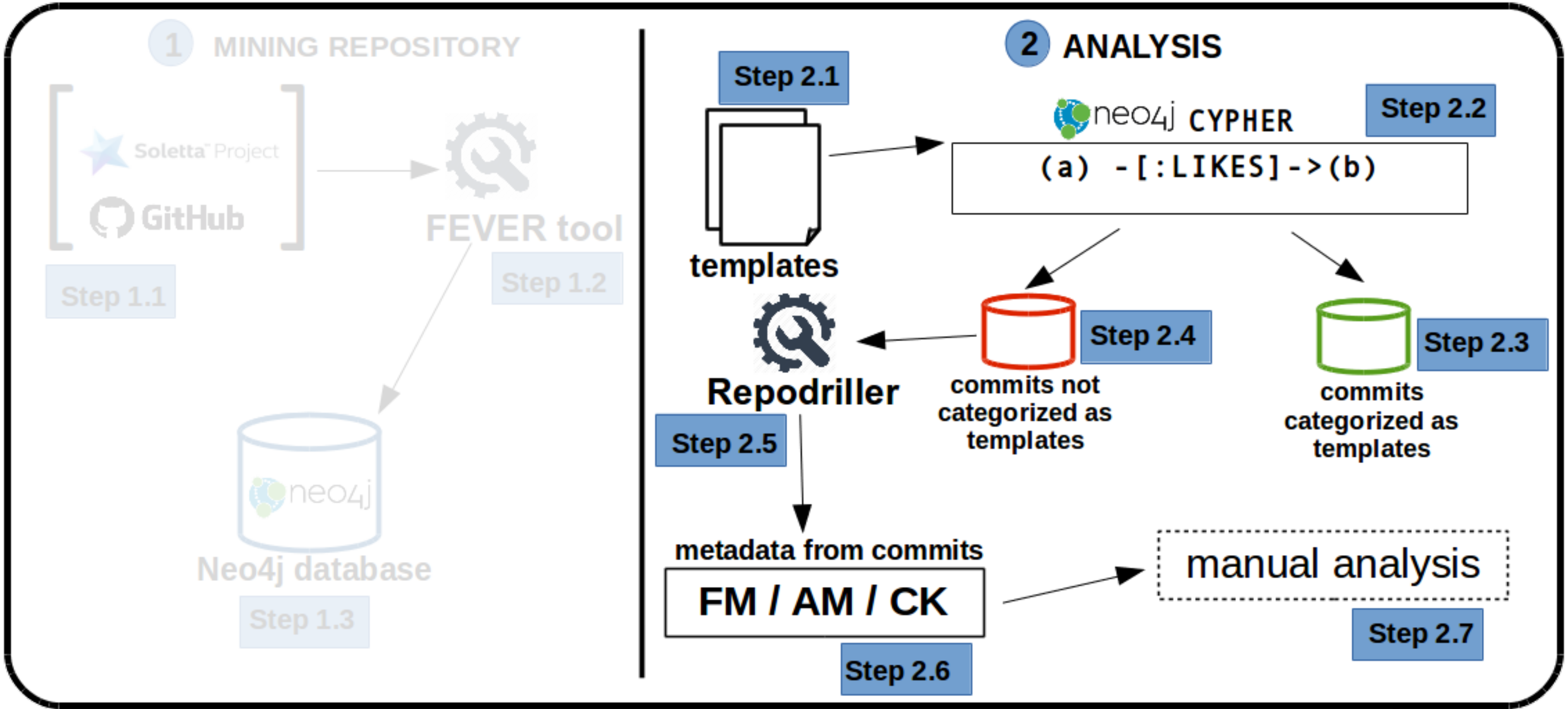
Centro de Informática UFPE
Universidade Federal de Pernambuco

Design geral do estudo (1. mineração)



[VaMoS'19, Karine's MSc dissertation]

Design geral do estudo (2. análise)



[VaMoS'19, Karine's MSc dissertation]

(muito) Breve resumo dos resultados para o Linux

Linux v3.12-3.13	Commits	%
Total	13,288	100%
Templates	11,377	85.62%
Not captured as Template	1,911	14.38%
Excluded	15	0.11%
Remaining Commits	1,896	14.27%

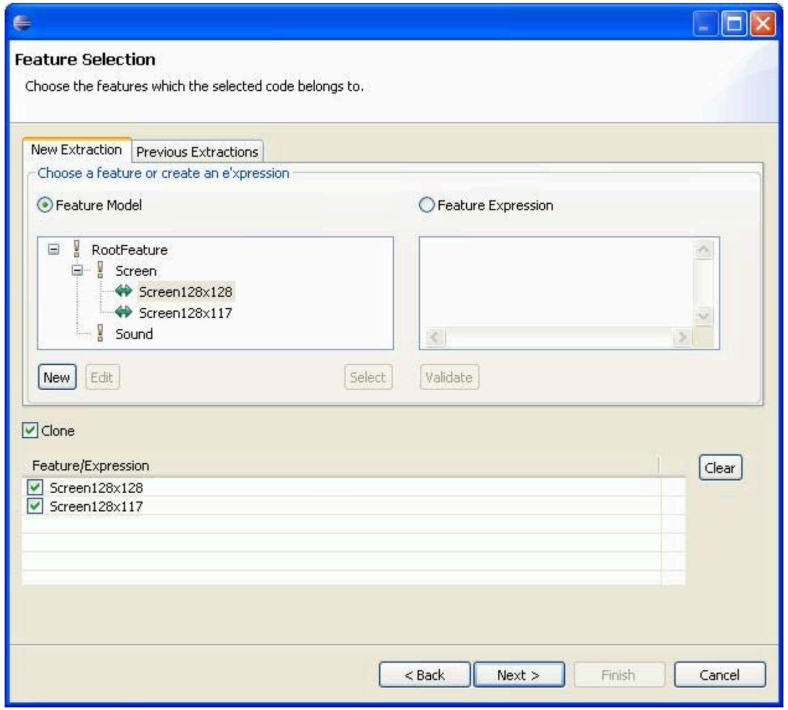
[VaMoS'19, Karine's MSc dissertation]

Desdobramentos

- Infraestrutura para 'explicar' mudanças ocorridas em um cenário de evolução
  - Parte disso equivale a identificar se evolução segura aconteceu
  - Também é útil como uma forma de análise de impacto de mudança, pode ajudar a identificar quais produtos testar e validar
- Também pode ajudar ferramentas de análise que verificam mudanças, como o SafeRefactor faz para verificar operações de refatoração de IDEs
- Série de outros trabalhos e orientações em temas relacionados e periféricos (slide a seguir)

Algumas lições...

Partimos de um problema e sua solução (concreta), para então formalizarmos uma teoria (abstrata)


Feature Selection

Choose the features which the selected code belongs to.

New Extraction | Previous Extractions

Choose a feature or create an expression

[✓ Feature Model] Feature Expression

- RootFeature
  - Screen
    - Screen128x128
    - Screen128x117
  - Sound

New Edit Select Validate

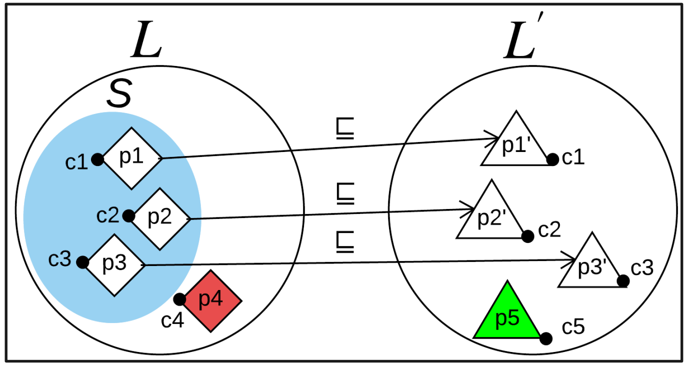
[✓] Clone

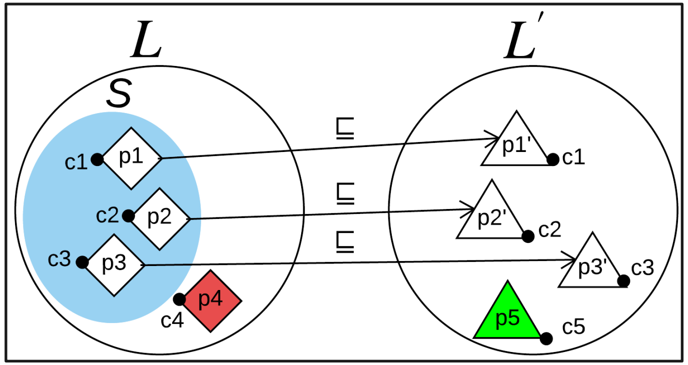
Feature/Expression

[✓] Screen128x128
[✓] Screen128x117

Clear

< Back Next > Finish Cancel





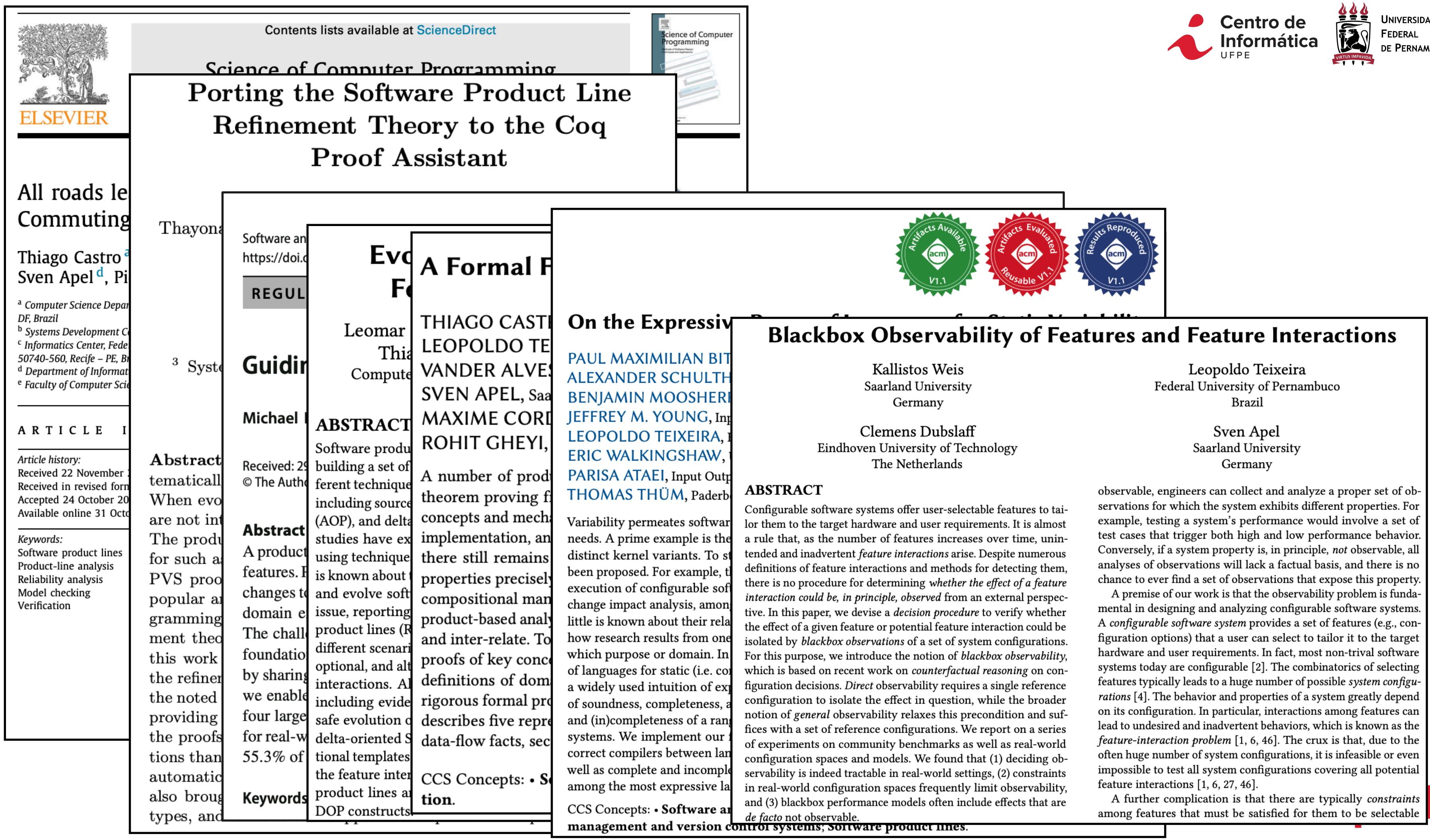
Não tenha medo de formalizar as coisas! Ajuda a consolidar ideias e entendimento!

Ideias levam tempo para amadurecer e serem disseminadas

(primeiro paper no começo dos anos 2000...)

Centro de Informática UFPE

Universidade Federal de Pernambuco



Contents lists available at ScienceDirect

Science of Computer Programming

ELSEVIER

All roads le
Commuting

Thiago Castro [a] Sven Apel [d], Pi

[a] Computer Science Depar DF, Brazil
[b] Systems Development C
[c] Informatics Center, Fede 50740-560, Recife – PE, Br
[d] Department of Informat
[e] Faculty of Computer Scie

A R T I C L E I

Article history:
Received 22 November
Received in revised form
Accepted 24 October 20
Available online 31 Octo

Keywords:
Software product lines
Product-line analysis
Reliability analysis
Model checking
Verification

Porting the Software Product Line Refinement Theory to the Coq Proof Assistant

Thayona

[3] Syst

Abstract
tematicall
When evo
are not int
The produ
for such a
PVS proo
popular ar
gramming
ment theo
this work
the refinem
the noted
providing
the proofs
tions than
automatic
also broug
types, and

Software an
https://doi.o

REGUL

Guidin

Michael

Received: 29
© The Autho

Abstract
A product
features.
changes to
domain e
The chall
foundatio
by sharing
we enable
four large
for real-w
55.3% of

Keywords

Evo
Fe

Leomar
Thia
Compute

ABSTRACT
Software produ
building a set of
ferent technique
including source
(AOP), and delta
studies have ex
using technique
is known about
and evolve soft
issue, reporting
product lines (R
different scenari
optional, and alt
interactions. Al
including evide
safe evolution o
delta-oriented S
tional templates
the feature inter
product lines a
DOP constructs

A Formal F

THIAGO CAST
LEOPOLDO TE
VANDER ALVE
SVEN APEL, Saa
MAXIME CORD
ROHIT GHEYI,

A number of prod
theorem proving f
concepts and mecha
implementation, an
there still remains
properties precisely
compositional man
product-based analy
and inter-relate. To
proofs of key conce
definitions of dom
rigorous formal pro
describes five repre
data-flow facts, sec

CCS Concepts: • So
tion.

Artifacts Available acm V1.1
Artifacts Evaluated acm Reusable V1.1
Results Reproduced acm V1.1

On the Expressiv

PAUL MAXIMILIAN BIT
ALEXANDER SCHULTH
BENJAMIN MOOSHER
JEFFREY M. YOUNG, Inp
LEOPOLDO TEIXEIRA,
ERIC WALKINGSHAW,
PARISA ATAEI, Input Outp
THOMAS THÜM, Paderb

Variability permeates softwar
needs. A prime example is the
distinct kernel variants. To st
been proposed. For example, t
execution of configurable sof
change impact analysis, amon
little is known about their rela
how research results from one
which purpose or domain. In
of languages for static (i.e. co
a widely used intuition of exp
of soundness, completeness, a
and (in)completeness of a rang
systems. We implement our
correct compilers between lan
well as complete and incomple
among the most expressive la

CCS Concepts: • Software ar
management and version control systems; Software product lines.

Blackbox Observability of Features and Feature Interactions

Kallistos Weis Saarland University Germany	Leopoldo Teixeira Federal University of Pernambuco Brazil
Clemens Dubslaff Eindhoven University of Technology The Netherlands	Sven Apel Saarland University Germany

ABSTRACT

Configurable software systems offer user-selectable features to tailor them to the target hardware and user requirements. It is almost a rule that, as the number of features increases over time, unintended and inadvertent *feature interactions* arise. Despite numerous definitions of feature interactions and methods for detecting them, there is no procedure for determining *whether the effect of a feature interaction could be, in principle, observed* from an external perspective. In this paper, we devise a *decision procedure* to verify whether the effect of a given feature or potential feature interaction could be isolated by *blackbox observations* of a set of system configurations. For this purpose, we introduce the notion of *blackbox observability*, which is based on recent work on *counterfactual reasoning* on configuration decisions. *Direct* observability requires a single reference configuration to isolate the effect in question, while the broader notion of *general* observability relaxes this precondition and suffices with a set of reference configurations. We report on a series of experiments on community benchmarks as well as real-world configuration spaces and models. We found that (1) deciding observability is indeed tractable in real-world settings, (2) constraints in real-world configuration spaces frequently limit observability, and (3) blackbox performance models often include effects that are *de facto* not observable.

observable, engineers can collect and analyze a proper set of observations for which the system exhibits different properties. For example, testing a system's performance would involve a set of test cases that trigger both high and low performance behavior. Conversely, if a system property is, in principle, *not* observable, all analyses of observations will lack a factual basis, and there is no chance to ever find a set of observations that expose this property.

A premise of our work is that the observability problem is fundamental in designing and analyzing configurable software systems. A *configurable software system* provides a set of features (e.g., configuration options) that a user can select to tailor it to the target hardware and user requirements. In fact, most non-trival software systems today are configurable [2]. The combinatorics of selecting features typically leads to a huge number of possible *system configurations* [4]. The behavior and properties of a system greatly depend on its configuration. In particular, interactions among features can lead to undesired and inadvertent behaviors, which is known as the *feature-interaction problem* [1, 6, 46]. The crux is that, due to the often huge number of system configurations, it is infeasible or even impossible to test all system configurations covering all potential feature interactions [1, 6, 27, 46].

A further complication is that there are typically *constraints* among features that must be satisfied for them to be selectable

Encontre bons colaboradores!

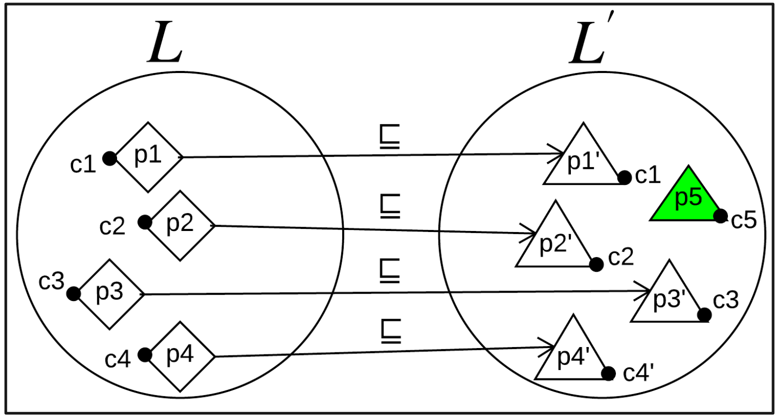
(faz parte do processo se divertir durante...)

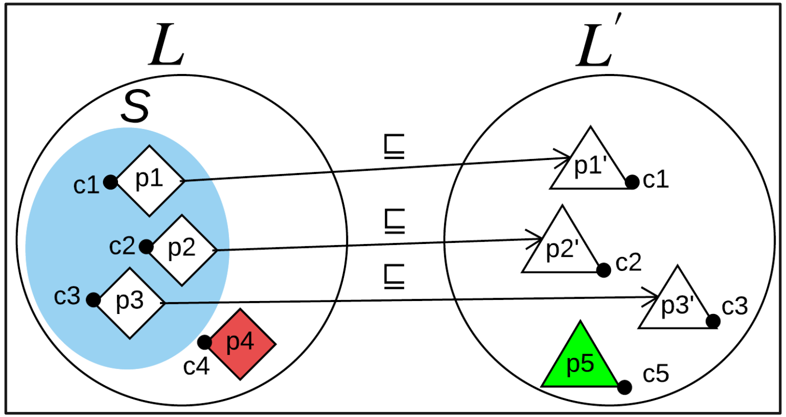
With a ~~little~~ lot of help from my friends...

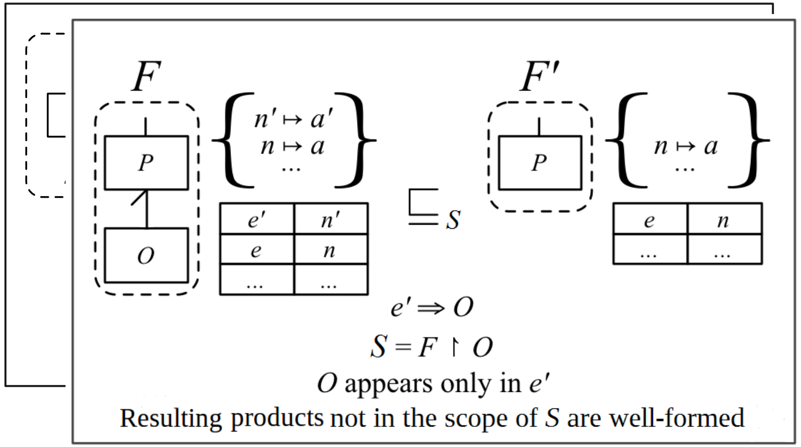
- Paulo Borba, Gabriela Sampaio, Karine Gomes, Thayonara Alves (UFPE)
- Vander Alves, Thiago Castro (UnB)
- Rohit Gheyi, Melina Mongiovi (UFCG)
- Márcio Ribeiro (UFAL)
- Uirá Kulesza (UFRN)
- Sven Apel, Kallistos Weis (Universität des Saarlandes)
- Michael Nieke, Ina Schaefer (TU Braunschweig)
- Thomas Thüm, Paul Bittner (Universität Ulm & Paderborn)
- Christoph Seidl (ITU Copenhagen)
- Maxime Cordy (Université du Luxembourg)

Centro de Informática UFPE

Universidade Federal de Pernambuco







e'	n'
e	n
...	...

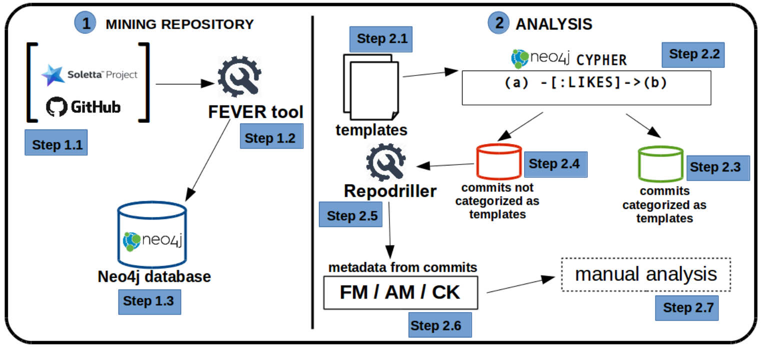
e	n
...	...

$$e' \Rightarrow O$$

$$S = F \upharpoonright O$$

O appears only in e'

Resulting products not in the scope of S are well-formed



1 MINING REPOSITORY
Soletta Project
GitHub
Step 1.1
FEVER tool
Step 1.2
Neo4j database
Step 1.3
2 ANALYSIS
Step 2.1
templates
neo4j CYPHER
Step 2.2
(a) -[:LIKES]->(b)
Repodriller
Step 2.5
commits not categorized as templates
Step 2.4
commits categorized as templates
Step 2.3
metadata from commits
FM / AM / CK
Step 2.6
manual analysis
Step 2.7