Low-level Variability Support for Web-based Software Product Lines

Ivan Machado[1,2], Alcemir Santos[1,2], Yguaratã Cavalcanti[2], Eduardo Trzan[3], Marcio Souza[3], Eduardo Almeida[1,2]

[1]Federal University of Bahia, Brazil (UFBA)
[2]Reuse in Software Engineering, Brazil (RiSE)
[3]Recôncavo Institute of Technology (IRT)
Salvador – Bahia – Brazil

We face a growing number of web-ready devices

MULTIPLE

DEVICES, PLATFORMS, O/S VERSIONS

...

… and a need to accommodate such a number of client-side specifications!

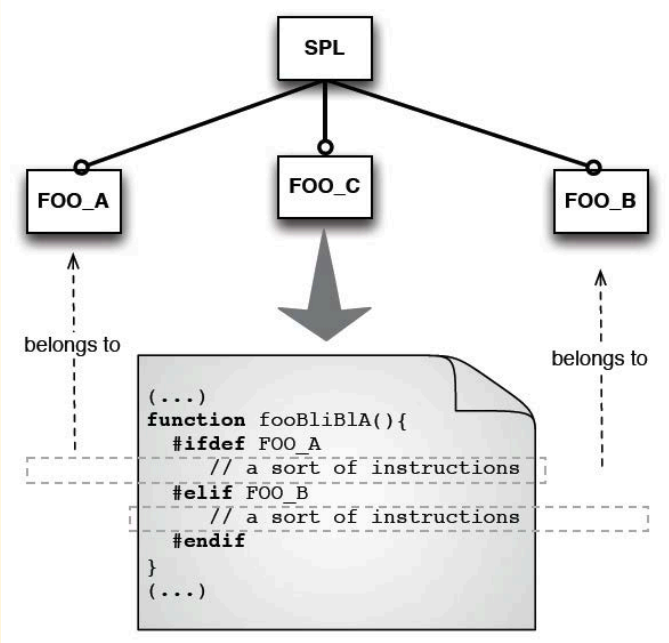
That is, a same web application should run properly in as many devices/platforms/versions as possible!

without increasing software development cost and time!

Variability Management matters!!!!

Expected behavior also matters!

- A given functionality in a web system can behave differently depending on, e.g., **accessibility** or **security requirements**, and the system is also expected to work properly
  - Ex: **Feature C** should behave differently depending on the selection of either **Feature A** or **Feature B**.

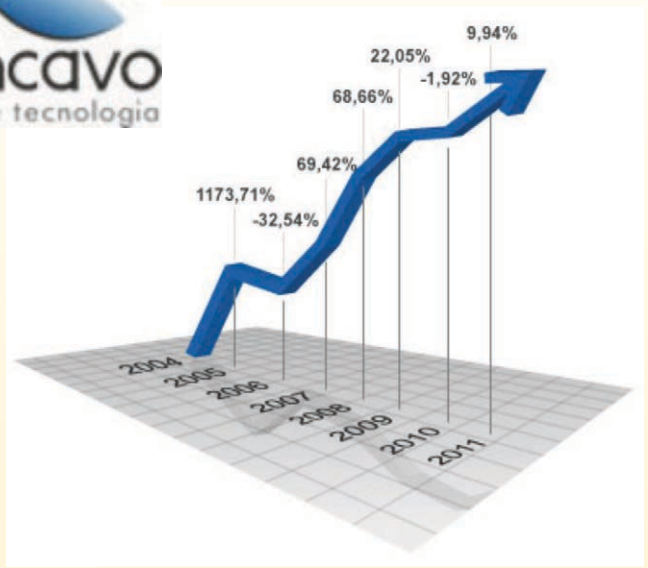


Motivation

- This is the scenario this experience report talks about:
  - Handling a web-based project to cope with variability, in a not so conventional programming language (at least for SPL engineering Literature)
  - JavaScript (HTML5)

Industry/Academia partnership

- One of the biggest Private R&D Institutes in Brazil
- The office is located in Salvador, Brazil
- Founded in 2004
- 120+ employees
- Innovation projects
- Third-party contractor for big players in the Brazilian Software market



Source: http://www.reconcavo.org.br/quem-somos/

Industry/Academia partnership

- To develop Interactive Learning Objects
  - Web-based systems intended for K-12 education
  - Interactive content

Industry/Academia partnership

Source: http://www.positivoteceduc.com.br/categoria/aplicativos-educacionais/

Interactive Learning Objects

- Web-based systems
- Shares functionalities
- Varies between “sub-domains”:
  - e.g. Science, Math, History, etc.
- Objects in a same “sub-domain“ only differ in their metadata
- **A likely scenario for SPL Engineering**

How things WERE done prior to SPL

- Learning Objects (Apps) were developed one at a time
- They were developed in the ActionScript language, until they decided to turn them platform-independent
- JavaScript (HTML5) was chosen as a candidate language
- **But at this point some problems emerged!**
  - **Reuse was merely opportunistic**
  - **Costs vs. Scale**
  - **Several existing (evolvable) APPs to porting from AS to JS**

Transitioning to SPL Engineering

- As the company was looking for a strategy to cope with such a demand
- We could convince their managers what SPL/FOSD was good for, and that it could fit the company's demand, indeed
- And… a feasible way we found to make its engineers aware of SPL was to apply it in a real project

What the SPL literature tells us about?

- Strong focus on Embedded Systems, Information Systems
  - Modeling techniques, Tools, Variability implementation mechanisms etc.
- Just a few reports on SPLs in the **Web systems domain**
  - Mostly covering **variability modeling aspects**, rather than discussing the role of variability at a lower level of abstraction, i.e., at **source code** level

Particular need of tool support

- Emerging need of tools to support variability management at **diverse abstraction levels**, to handle the particularities of different Web-based languages.
- They should cope with **feature interaction**, to accommodate the dynamic behavior of features
- Recall that Web systems often provide dynamic adaptive content to support different platforms and devices.

Our Proposal

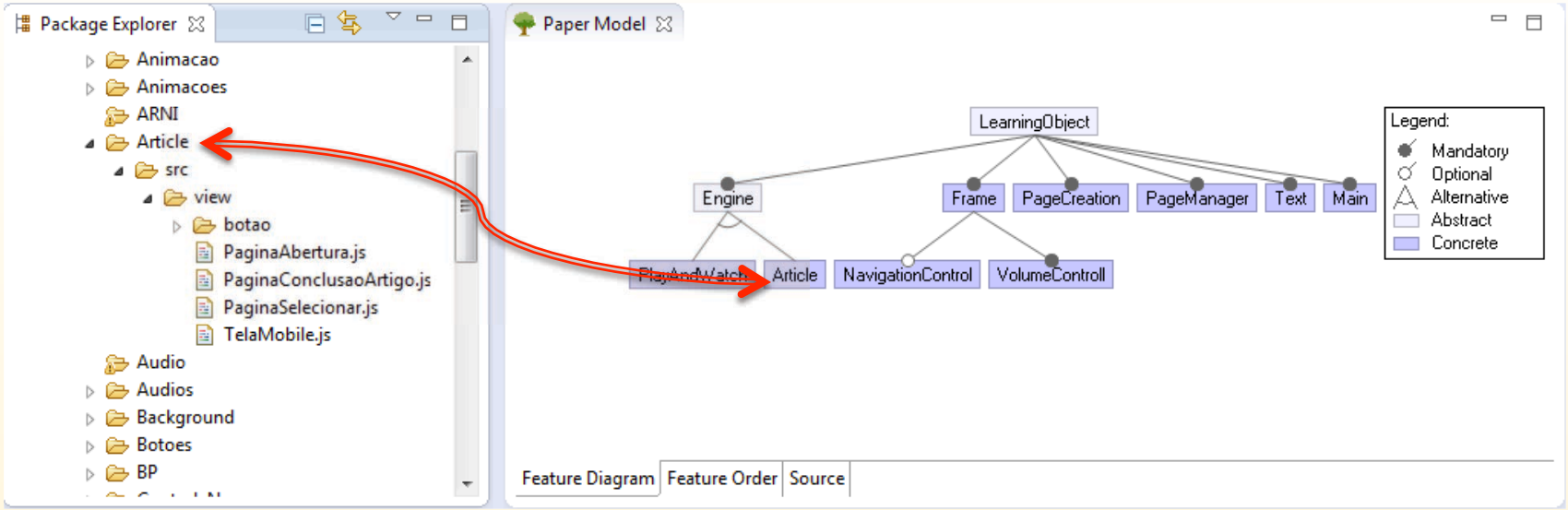
- The FeatureJS Plugin
  - Eclipse-based plugin to enable the development of SPLs in JavaScript
  - Extends the FeatureIDE (released under L-GPL license v3)
  - Plugin source code is available for download
  - A blending of **composition-based** development and **annotation-based** development

How FeatureJS works

- From FeatureIDE: **Feature Dependencies**
  - Enables the representation of constraints between features, controlled by the **configuration view**.
  - A **configuration** either enables or disables the selection of a given feature, according to the constraints associated to it.

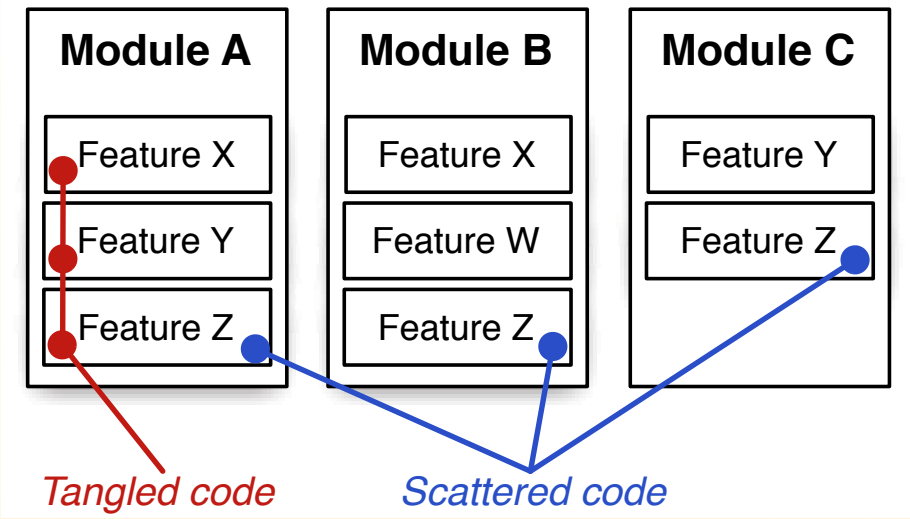
How FeatureJS works

- From FeatureIDE: **Containment Hierarchy**



How FeatureJS works

- Assuming that **feature interactions** also occur at source code level, and a single feature can be mapped to multiple code fragments.
- In practice, a feature does not map cleanly to an isolated module of code, but instead it may affect many artifacts



How FeatureJS works

- FeatureIDE partially controls variability at implementation level (to our scenario)
  - **Refinement declarations** (for languages like Java)
- However, for languages such as JavaScript, that do not enable those declarations, applying such a technique to control ***inner-function variability*** would lead to a large amount of duplicate code.

How FeatureJS works

- The applied solution
  - Mix of the (native FeatureIDE) **composition** with **annotation**
  - It enables variability management at implementation level.
  - **Composition**: handles the inclusion or exclusion of an entire function in a product variant
  - **Annotation**: enables inner-function statements behave differently, depending on the selection of a given feature

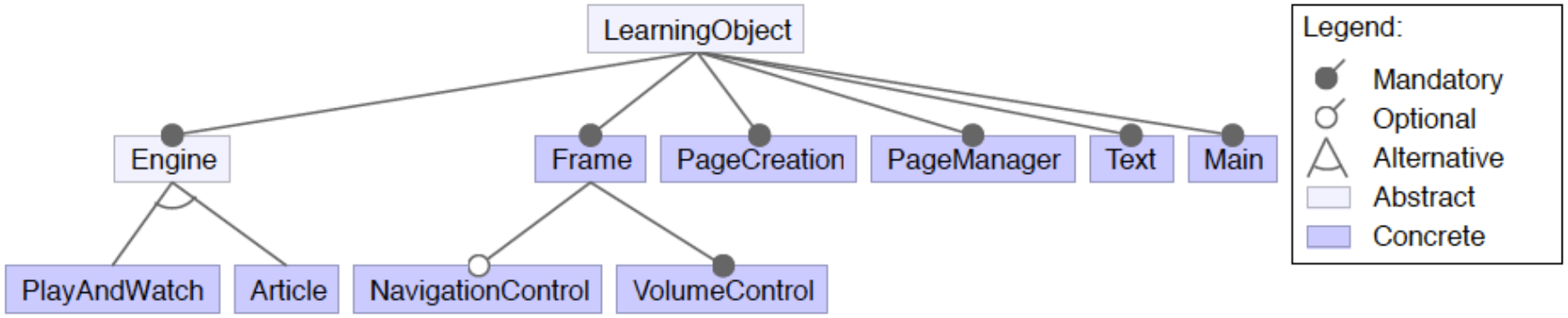
Down side of annotations

- Among different types of variability representation, **preprocessor directives** are often used to implement software variation because of simplicity and flexibility.
- However, its **obtrusive syntax** and **lack of structure** may reduce comprehensibility and increase maintenance costs.

So, why using annotations?

- Composition rules for augmenting functions with new properties in JavaScript is not always safe
- In cases more than one product configuration includes the same JavaScript file, but depending on the feature selection a function behaves differently, the use of preprocessor directives may be employed to generate different product variants.

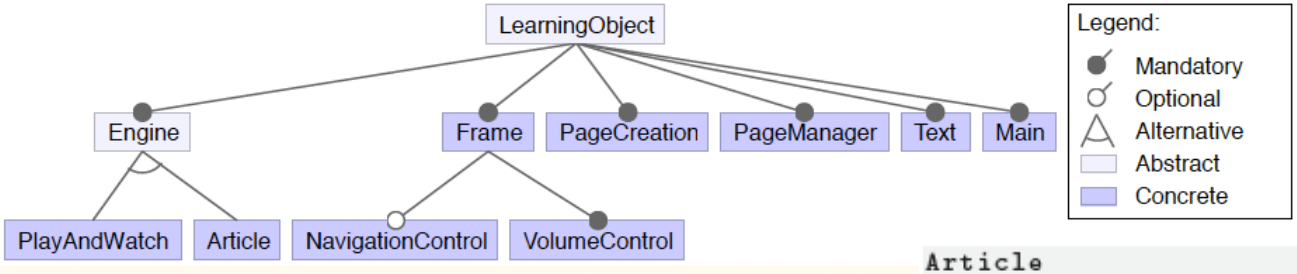
How FeatureJS works




```
(function (window){
                    :
// #ifdef PlayAndWatch
ref.clickForwardPage = function(event){
   if(_mode == _pageTypeEnum.PLAY || _mode ==
 _pageTypeEnum.PLAY_ANDROID || _mode ==
 _pageTypeEnum.PLAY_ANDROID_CHROME) { _page.
     animate();
   }
   else{
      _pageManager.forwardPage();
   }
}
// #elif Article
ref.clickForwardPage = function(event){
   _pageManager.forwardPage();
}
// #endif
                    :
}(window))
```

How FeatureJS works

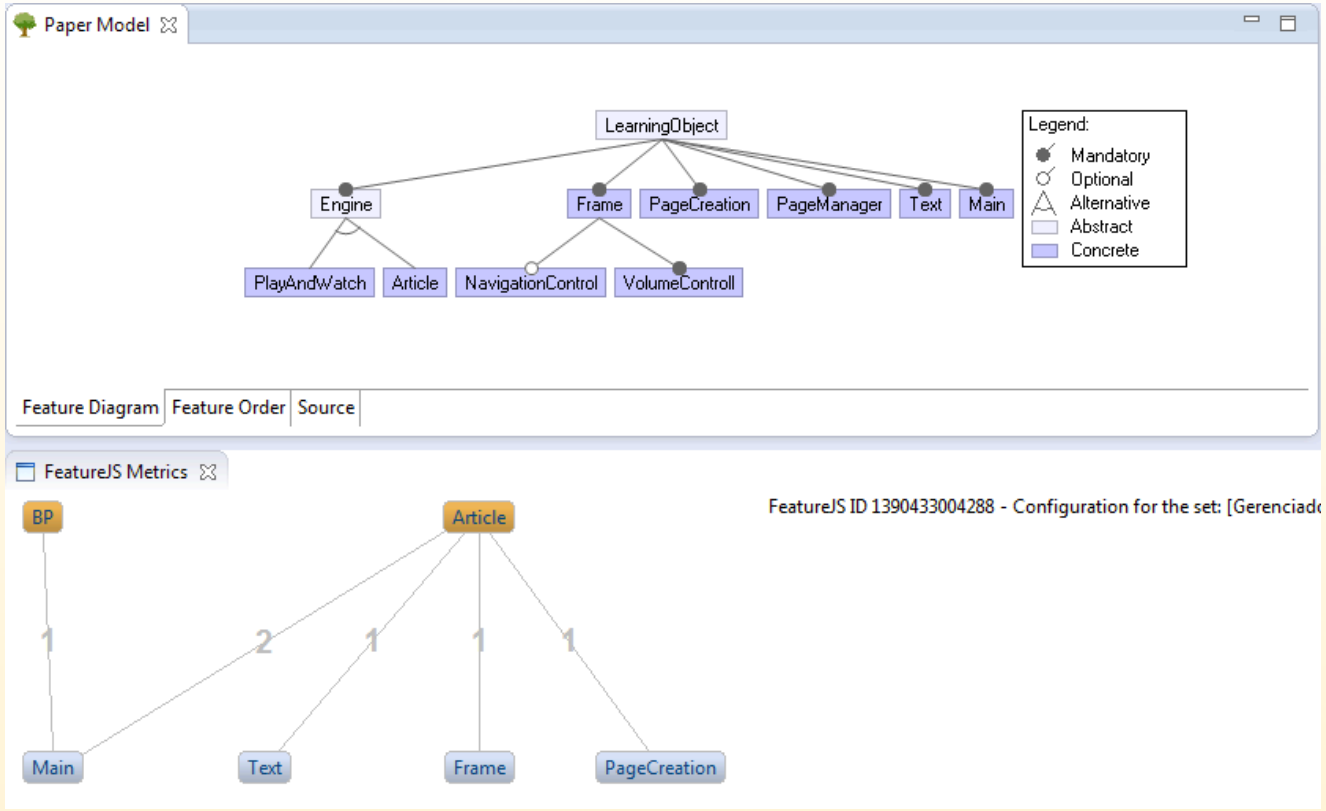
- A textual representation of Feature Interaction to improve comprehesibility



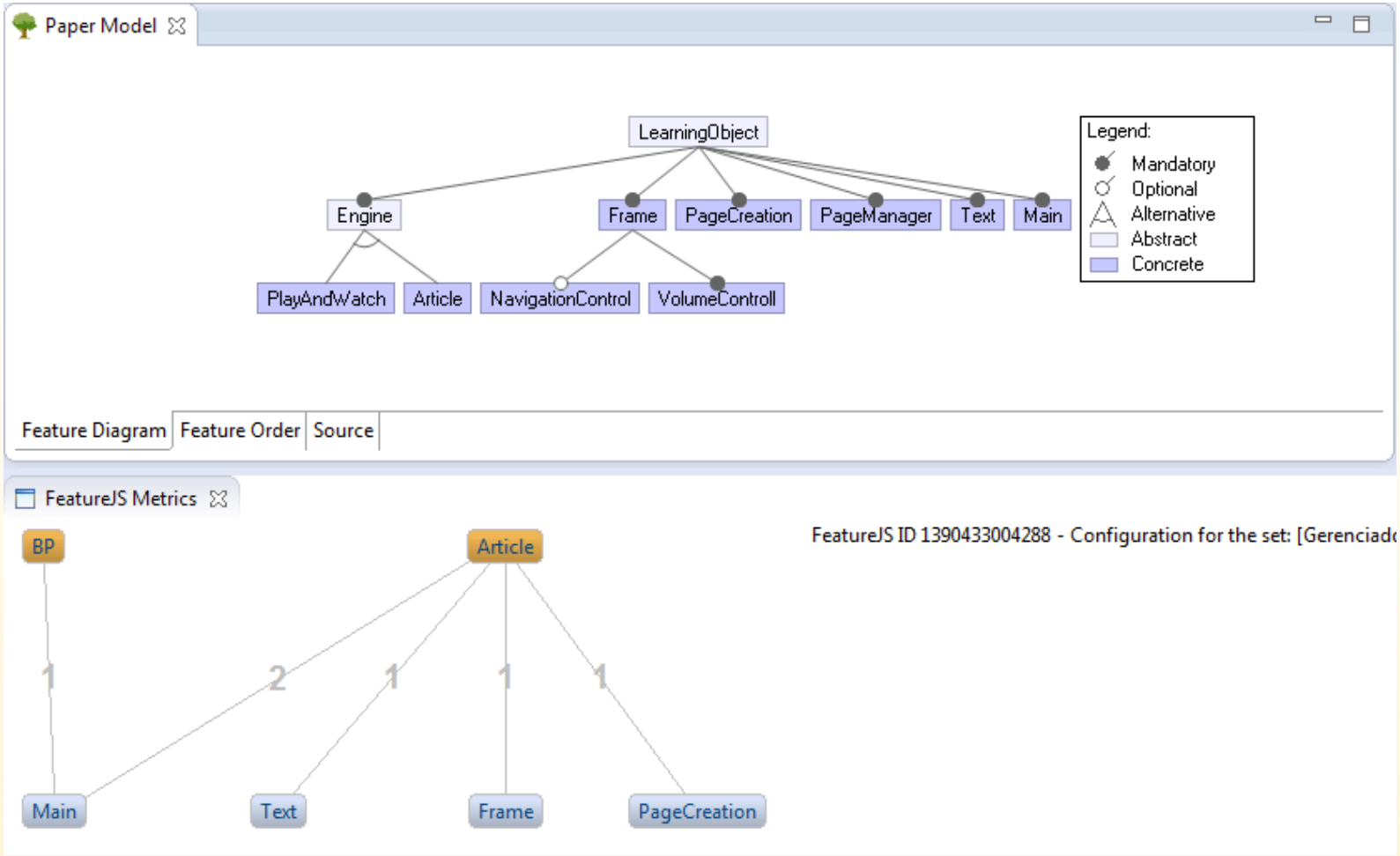

```
Article
- PageCreation: 1 (PageFactory.js),
- Main: 2 (Main.js, index.html),
- Frame: 1 (Frame.js),
- Text: 1 (FabricaTextoLegenda.js);

BP
- Main: 1 (index.html).

ACJC
- Main: 2 (Main.js, index.html);

PlayAndWatch
- PageCreation: 1 (PageFactory.js),
- PageManager: 1 (PageManager.js),
- Frame: 1 (Frame),
- Main: 2 (Main.js, index.html),
- Text: 1 (SubtitleFactory.js).
```

How FeatureJS works

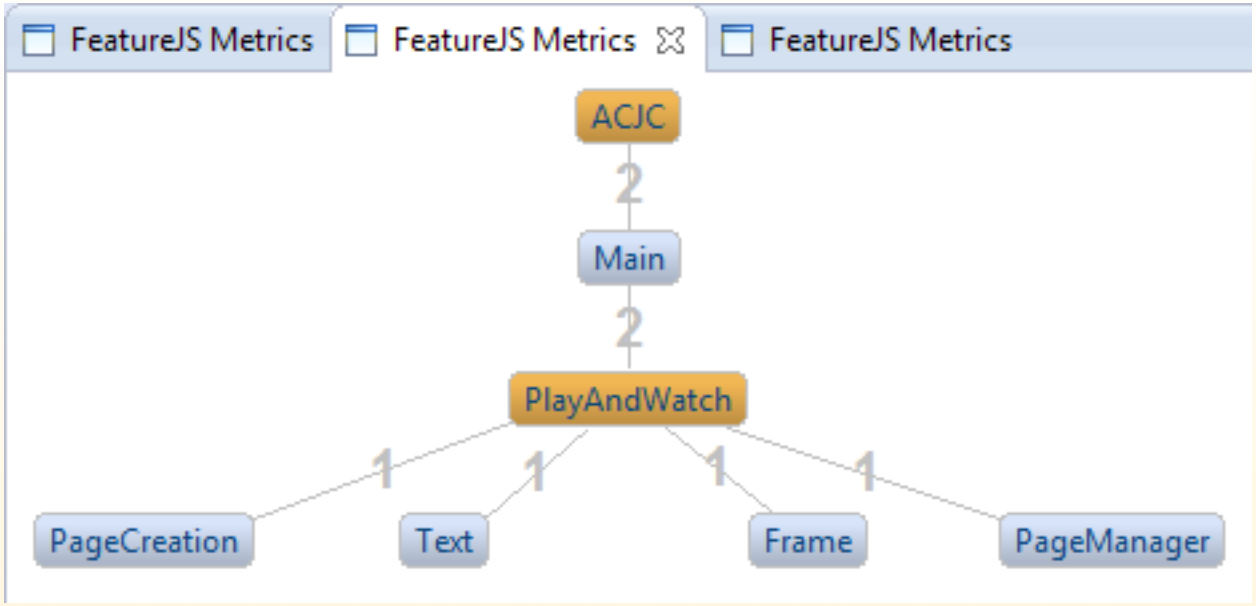
- A visual representation of Feature Interaction to improve comprehesibility (refactoring says YEAH!)



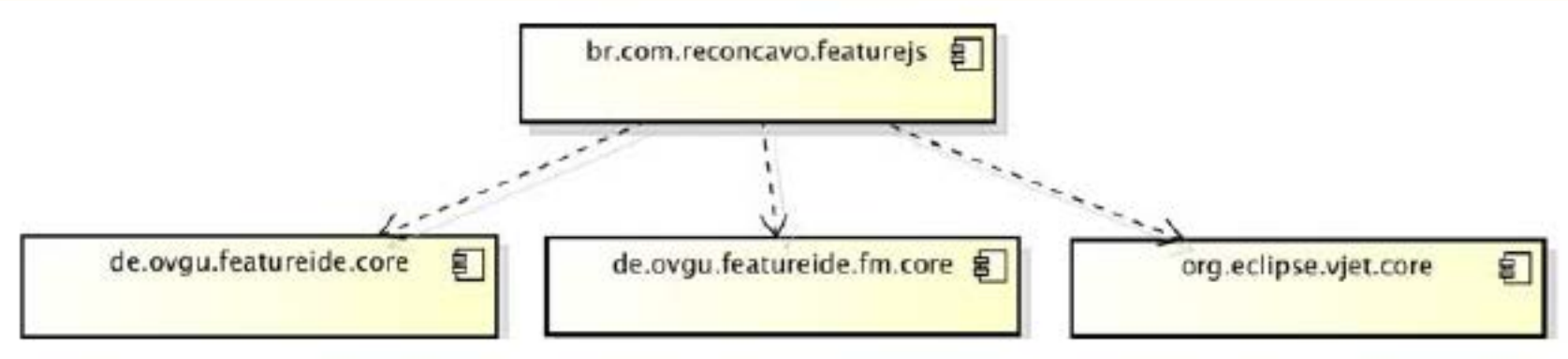
How FeatureJS works



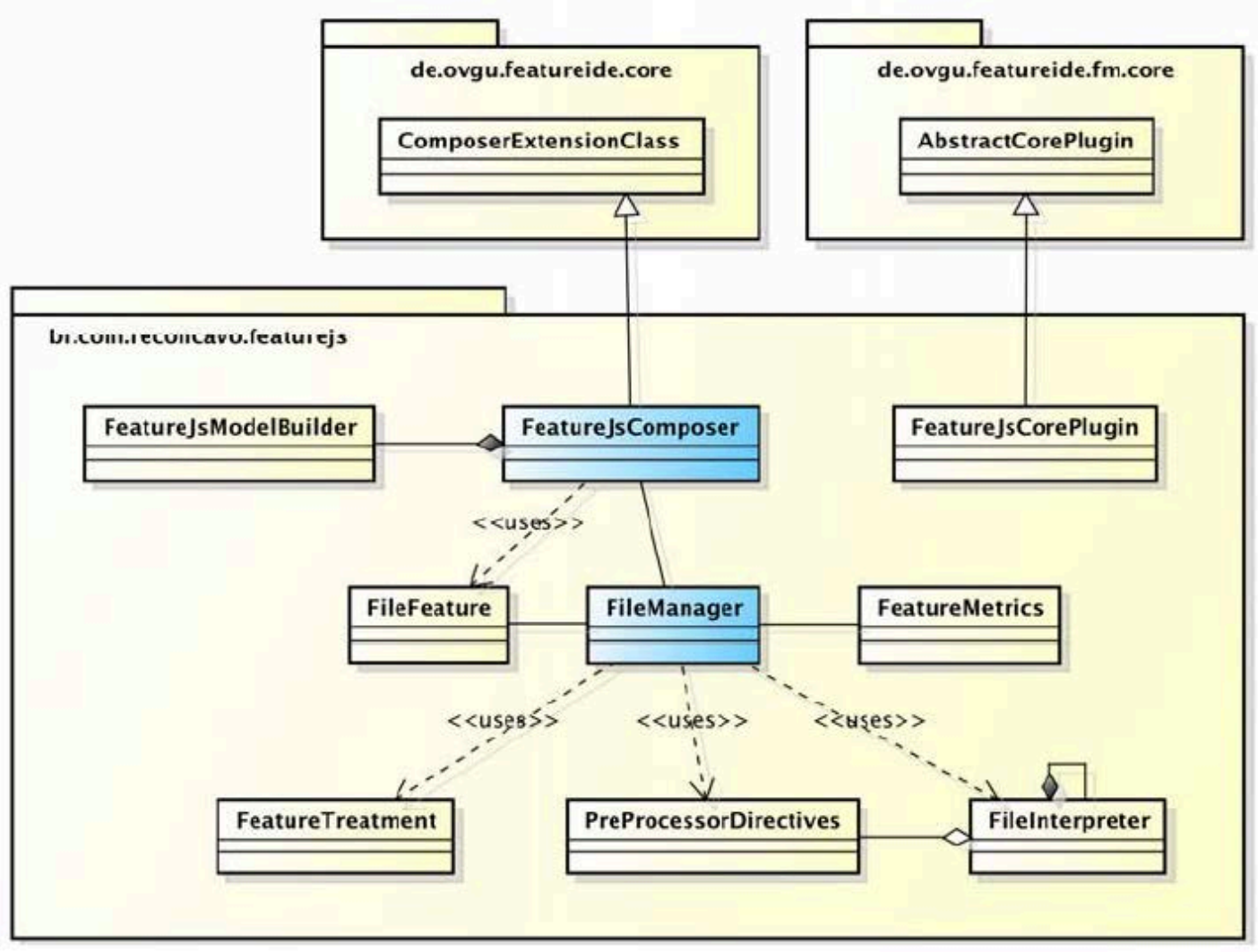
How FeatureJS works



FeatureJS Archictecture



FeatureJS Archictecture



FeatureJS Archictecture

- **VJET plugin:** supports JavaScript faster development, such as code completion, code templates, wizards, debug support, and native **type** and syntax **checking**, to identify errors through semantic validation.

- **ZEST Eclipse vizualization toolkit**: graphical editing framework

Preliminary Evaluation

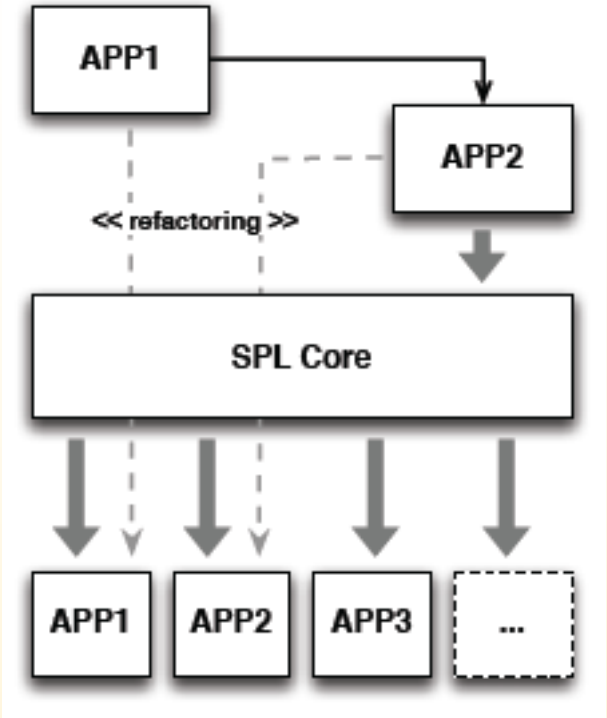
- Application Domain
  - Learning Objects (Web-based systems)

- MDC Learning Objects, comprises a set of 42 features. The core features has, together, around 3.7 KLOC.
- The MDC project has 23 boolean configuration variables and can, in theory, be deployed in over 3800 different configurations.
- Such number is not realistic in practice (Let's see why)

The project – cont'd

- The product-specific features mainly include the management of **metadata,** such as the media scripts, particular to every single learning object, and as such cannot be shared with other objects at all.
- Thus, for this particular case study we consider three different products, fully functional, generated from the core asset base.

Reactive SPL



Products metrics generated from the SPL

	LOC	Files	Functions	DS	ES
Core	3,778	47	421	796	2,003
APP1	5,568	62	510	972	3,243
APP2	5,188	61	518	964	3,039
APP3	6,520	63	514	978	4,027

DS: Declarative Statements, ES: Executable Statements.

Application	Dev. Time
APP1	720 engineer-hours
APP2 + SPL Core	448 engineer-hours
APP1 Refactoring	160 engineer-hours
APP3	122 engineer-hours

#	MDC Features	APP1	APP2	APP3
1	PageCreation	✔	✔	✔
2	WatchPage	✔		✔
3	PlayPage	✔		✔
4	ArticlePage		✔	
5	MatchColsTask		✔	
6	FillInTask		✔	
7	SubtitleManager	✔	✔	✔
8	VideoManager			
9	AnimationManager	✔	✔	✔
10	AudioManager	✔	✔	✔
11	NavigationControl	✔	✔	✔
12	Article		✔	
13	BP		✔	
14	PlayAndWatch	✔		✔
15	ACJC	✔		
16	AGR			✔
17	Animations	✔	✔	✔
18	Background	✔	✔	✔
19	Buttons	✔	✔	✔
20	Environment	✔		✔
21	Locutions	✔	✔	✔
22	Music	✔		✔
23	Effects	✔	✔	✔

✔: Selected feature.

Case study results

- Some gains in development time, what might result in cost reductions in next products' releases.
- However…
  - Limited evidence does not allow generalizations
  - The main gathered data refers to the development time as a function of the product variant size.
  - No previous data (baseline) available => the reduction in development time for the *n*-ary variant releases might be caused by a maturation effect.
- Hence, further studies are required!

Concluding remarks

- **FeatureJS:** provides automated support to SPL development in JavaScript and HTML
- **Blending of composition and annotation:** variability management at implementation level
- **Case study:** the plugin may positively impact both development cost and time, and the maintainability of the SPL (in this unusual SPL domain)

Concluding remarks

- **Type checking:** for JavaScript files, VJET fits. However, we do not handle yet preprocessor directives checking (must do)
- **Why not using commercial tools?**

Thank you for your attention!

Low-level Variability Support for Web-based Software Product Lines

Ivan Machado | ivanmachado@dcc.ufba.br

[1]Federal University of Bahia, Brazil (UFBA) | **Salvador – Bahia – Brazil**