Micro Pattern in Agile Software XP 2013, Wien

Giulio Concas, Giuseppe Destefanis, Michele Marchesi, Marco Ortu and Roberto Tonelli

Department of Electrical and Electronic Engineering (DIEE) of the University of Cagliari

06-06-2013

Introduction

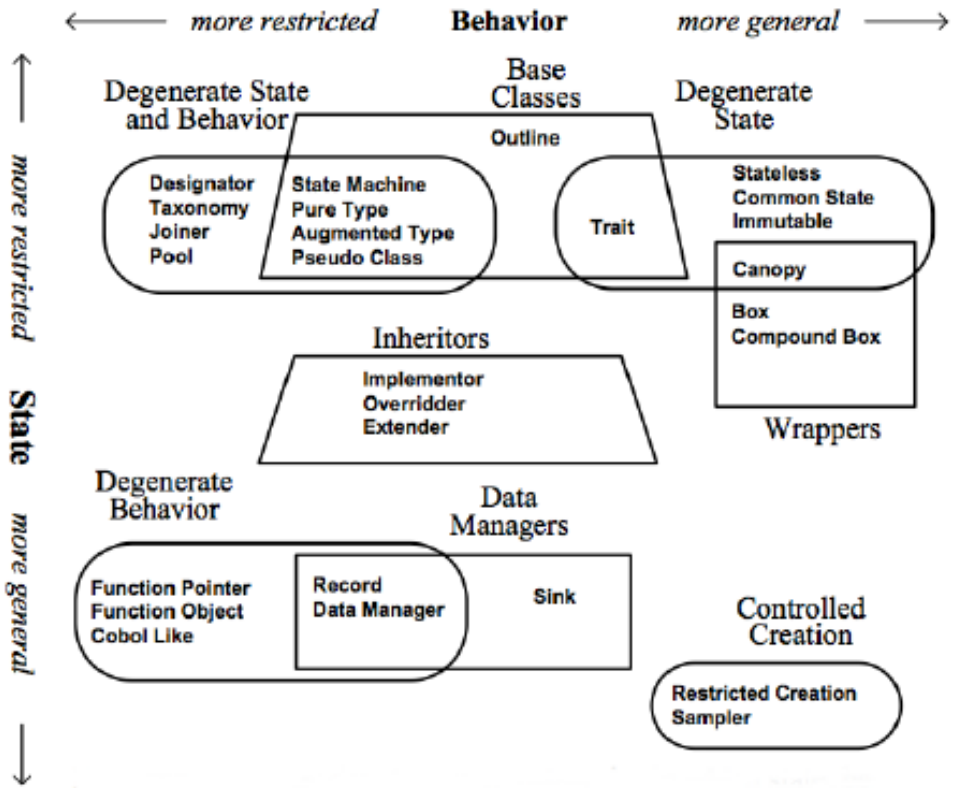
- Micro pattern are similar to design pattern but are at a lower level of abstraction with respect to design pattern, their characteristic is that they can be identified automatically.

Introduction

- Micro pattern are similar to design pattern but are at a lower level of abstraction with respect to design pattern, their characteristic is that they can be identified automatically.

- Gil and Maman defined the micro pattern catalogue that consists in 27 micro pattern for JAVA code.

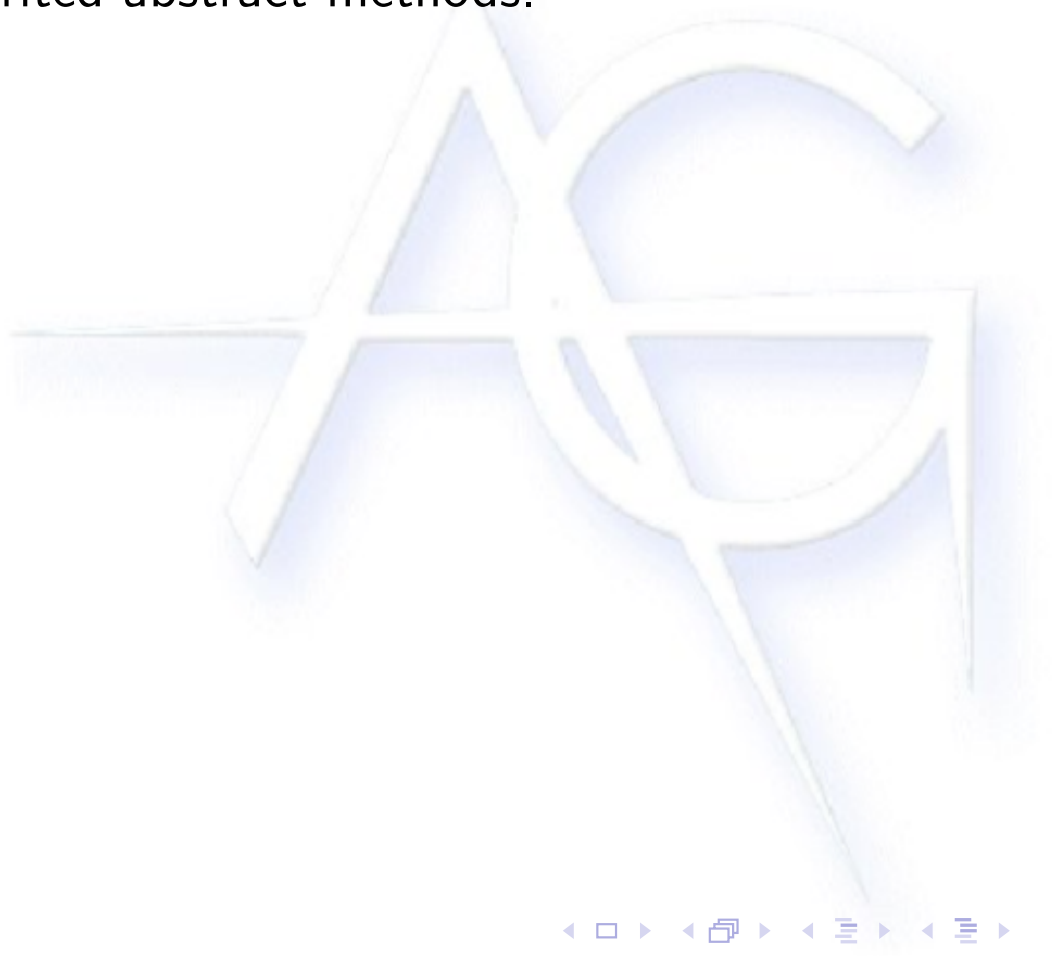
Introduction: MP catalogue



Gil-Maman Micro pattern catalogue

Introduction: MP Examples

- Implementor:
  - A concrete class, where all the methods override inherited abstract methods.



Introduction: MP Examples

- Implementor:
  - A concrete class, where all the methods override inherited abstract methods.
- Sink:
  - A class whose methods do not propagate calls to any other class.

Introduction: MP Examples

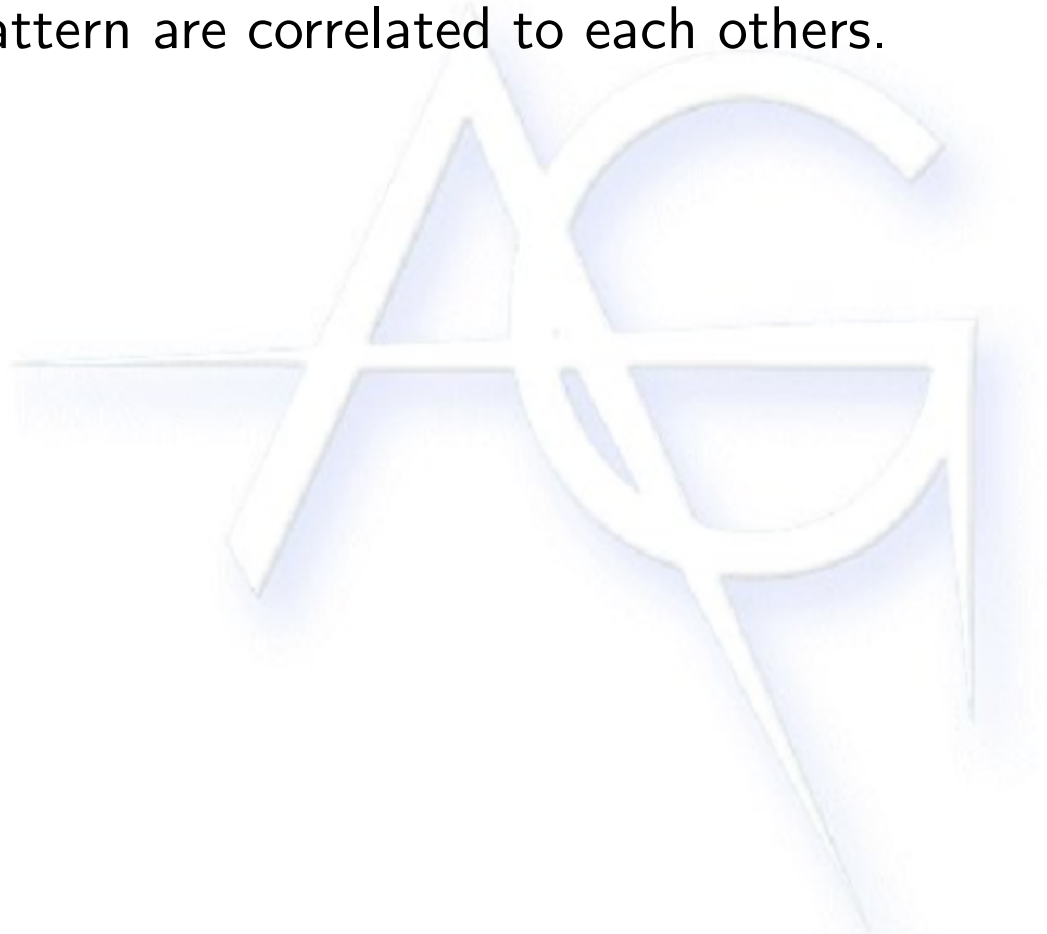
- Implementor:
  - A concrete class, where all the methods override inherited abstract methods.
- Sink:
  - A class whose methods do not propagate calls to any other class.
- Function Pointer:
  - A class with a single public instance method, but with no fields.

Introduction

- Gil and Maman state that at least 75% of classes belong to at least one micro pattern.

Introduction

- Gil and Maman state that at least 75% of classes belong to at least one micro pattern.
- Micro pattern are correlated to each others.



Introduction

- Gil and Maman state that at least 75% of classes belong to at least one micro pattern.
- Micro pattern are correlated to each others.
- Micro pattern can by categorized.

Introduction

- Gil and Maman state that at least 75% of classes belong to at least one micro pattern.
- Micro pattern are correlated to each others.
- Micro pattern can by categorized.
- Some micro pattern are more fault prone than others.

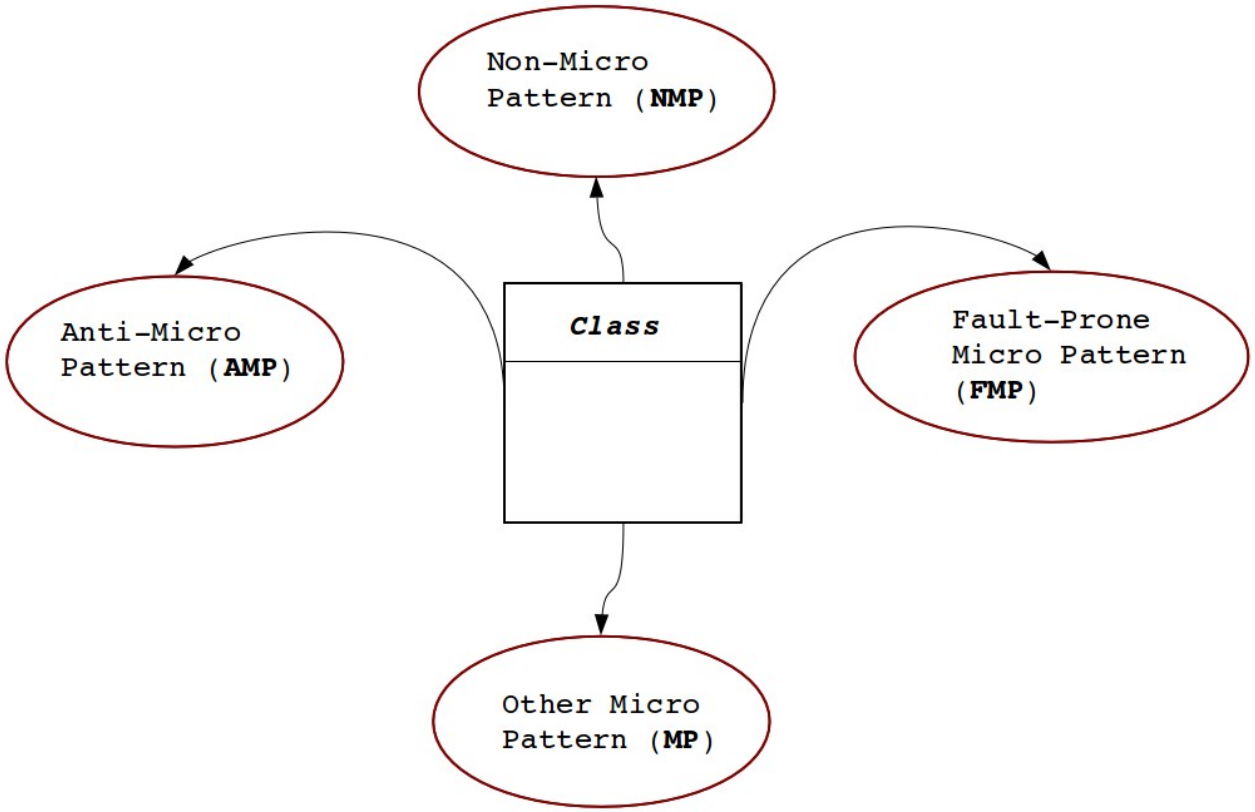
Introduction

- Gil and Maman state that at least 75% of classes belong to at least one micro pattern.
- Micro pattern are correlated to each others.
- Micro pattern can by categorized.
- Some micro pattern are more fault prone than others.
- Anti-micro pattern are associated to poor programming practices.

Introduction

- Gil and Maman state that at least 75% of classes belong to at least one micro pattern.
- Micro pattern are correlated to each others.
- Micro pattern can by categorized.
- Some micro pattern are more fault prone than others.
- Anti-micro pattern are associated to poor programming practices.
- Non-micro pattern are proved to be more fault prone.

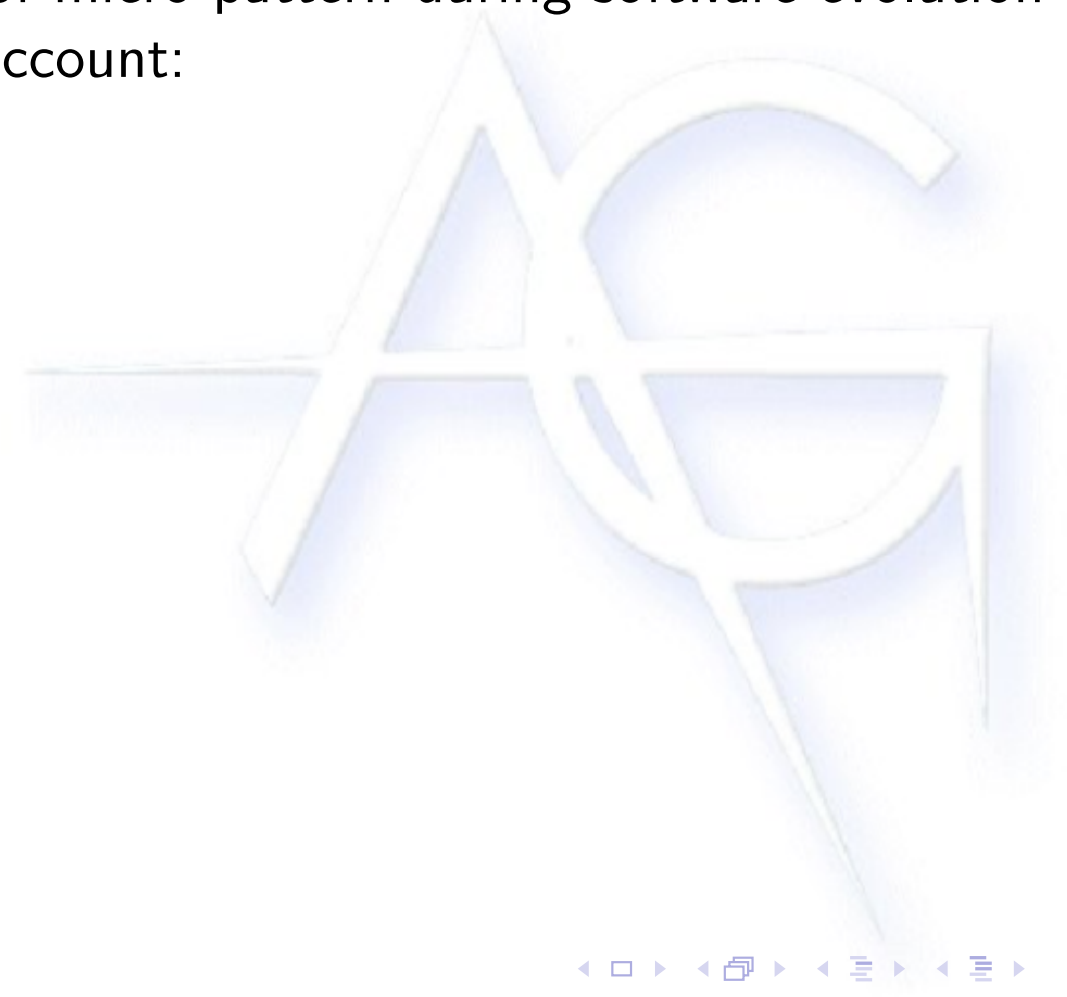
Introduction



Class categorization using micro pattern

Introduction

We investigated if agile methodologies influence the distribution of micro pattern during software evolution taking into account:



Introduction

We investigated if agile methodologies influence the distribution of micro pattern during software evolution taking into account:

- Anti-micro pattern and fault-prone micro pattern distribution evolution.

Introduction

We investigated if agile methodologies influence the distribution of micro pattern during software evolution taking into account:

- Anti-micro pattern and fault-prone micro pattern distribution evolution.
- Micro pattern fault analysis.

Introduction

We studied two industrial cases:

Giulio Concas, Giuseppe Destefanis, Michele Marchesi, Marco Ortu and Roberto Tonelli

Introduction

We studied two industrial cases:

- Floss-AR: web application for research publications management.

Introduction

We studied two industrial cases:

- Floss-AR: web application for research publications management.
- JAPS: java framework for enterprise portal building.

Methodology

We divided the activities of our studies in three main steps:

Methodology

We divided the activities of our studies in three main steps:

- 1. We proposed a survey to the development team to discover which agile methodologies were adopted.

Methodology

We divided the activities of our studies in three main steps:

- 1. We proposed a survey to the development team to discover which agile methodologies were adopted.
- 2. We mined source code repository to link classes to bugs.

Methodology

We divided the activities of our studies in three main steps:

- 1. We proposed a survey to the development team to discover which agile methodologies were adopted.
- 2. We mined source code repository to link classes to bugs.
- 3. Finally, we asses micro pattern fault proneness by means of faulty classes.

Methodology

Project	Micro pattern distribution evolution analysis	Micro pattern fault analysis	Survey
FLOSS-AR	YES	YES	YES
JAPS	YES	NO	NO

Analysis performed on the two system

Methodology: Micro pattern detection

- Arcelli and Maggioni suggest an operative definition to automate micro pattern detection.

Methodology: Micro pattern detection

- Arcelli and Maggioni suggest an operative definition to automate micro pattern detection.
- Based on their definition, we developed a tool to detect classes micro pattern.

Methodology: Bug-Class linking

Source code and bug fixes are linked using the Bachmann and Bernstein's heuristics :

Methodology: Bug-Class linking

Source code and bug fixes are linked using the Bachmann and Bernstein's heuristics :

- 1. Scan through the change logs for bug reports in a given format (e.g. fix bug,fix issue and so on).

Methodology: Bug-Class linking

Source code and bug fixes are linked using the Bachmann and Bernstein's heuristics :

- 1. Scan through the change logs for bug reports in a given format (e.g. fix bug,fix issue and so on).
- 2. Exclude all false positive bug numbers (e.g. r420, 2009-05-07 10:47:39 -0400 and so on).

Methodology: Bug-Class linking

Source code and bug fixes are linked using the Bachmann and Bernstein's heuristics :

- 1. Scan through the change logs for bug reports in a given format (e.g. fix bug,fix issue and so on).
- 2. Exclude all false positive bug numbers (e.g. r420, 2009-05-07 10:47:39 -0400 and so on).
- 3. Check if there are other potential bug number formats or false positive number formats, add the new formats and scan the change logs iteratively.

Methodology: Bug-Class linking

Source code and bug fixes are linked using the Bachmann and Bernstein's heuristics :

- 1. Scan through the change logs for bug reports in a given format (e.g. fix bug,fix issue and so on).
- 2. Exclude all false positive bug numbers (e.g. r420, 2009-05-07 10:47:39 -0400 and so on).
- 3. Check if there are other potential bug number formats or false positive number formats, add the new formats and scan the change logs iteratively.
- 4. Check if potential bug numbers exist in the bug-tracking database with their status marked as fixed.

Survey: Team Manager Interview

Survey video

Survey: Collaborations

Question	Very good	Good	Discrete	Adequate	Not adequate
How would you describe the collaboration of the team?	4	1	0	0	0

Survey's questions on collaboration

Survey: Developers

Question	Yes	No
The collaboration inside the team increased the productivity?	5	0
Did you develop the whole system?	3	2
Have the project decision been discussed together with the team?	5	0
Did you interact directly with the customer?	4	1
Did you use refactoring?	5	0

Survey's questions to the developers

Survey: Project Manager

Question	Answer
Which Agile methodologies did you use during development?	▶ Pair Programming ▶ Stand Up Meeting ▶ Refactoring ▶ On Site Customer
How often did you interact with the customer?	1-2 times per month
How often did you use refactoring?	2-3 times per month

Survey's questions to the project manager

Results

In this section we present the distribution of MP for the two analized projects.

Results: FLOSS-AR

MP	1.0	1.2	1.4	1.6	1.6.2	1.8	1.8.2	2.0
DESIGNATOR	2.14	1.79	2	3.3	3	4.32	6.83	9.6
TAXONOMY	0	0	0	0	0	0	0	0
POOL	0	0	0	0.55	0.54	0.27	0	0.35
JOINER	0	0	0	0	0	0	0	0
FUNCTIONPOINTER	27.1	23.3	27.5	18.7	19.5	18.1	16.7	7.18
FUNCTIONOBJECT	0.71	6.1	0	2.2	2.7	1.89	2.02	1.22
COBOLLIKE	0	0	0	0.27	0.27	0.81	0.75	0.5
STATELESS	0.71	0	1	0.82	0.82	1.08	1.01	1.22
COMMONSTATE	0	0	0	0	0	0	0	0.17
IMMUTABLE	0	3.2	0	0.82	0.82	0.81	0.75	0.87
RESTRICTEDCREATION	0.35	0.4	0.33	0.55	0.54	0.54	0.5	0.17
SAMPLER	0	0	0	0	0	0	0	0
BOX	4.64	15.4	3.98	0.27	0.27	0.27	0.25	1.4
COMPOUNDBOX	7.5	10	12.3	7.1	17.9	7.02	6.83	11.9

Results: FLOSS-AR

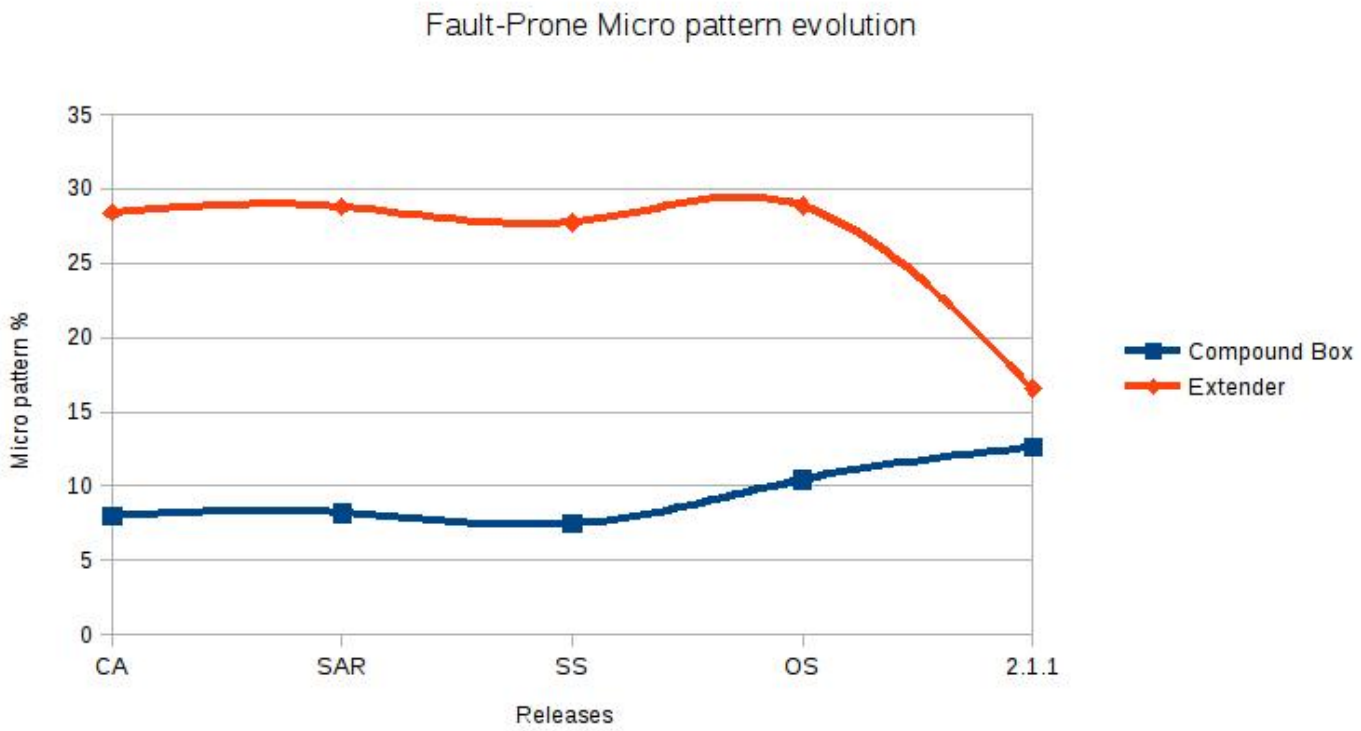
CANOPY	0	0	0	0	0	0	0	0
RECORD	0	0	0	0	0	0	0	0
DATAMANAGER	0.35	0.35	0	0	0	0	0	0
MP	1.0	1.2	1.4	1.6	1.6.2	1.8	1.8.2	2.0
SINK	15.3	3.9	15.6	4.14	3.5	2.7	2.78	2.45
OUTLINE	0	0	0	0	0	0	1.0	0.35
TRAIT	0	0	0	0	0	0	1.3	1.1
STATEMACHINE	0.71	0	0.66	0.82	0.82	0.54	0.5	5.4
PURETYPE	0	0	0	0	0.5	0.8	0.3	0.2
AUGMENTEDTYPE	0	0	0	0	0	0	0	0
PSEUDOCLASS	0	0	0	0	0	0	0	0
IMPLEMENTOR	0	0.71	0.3	0.27	0.27	0.27	0.25	0.35
OVERRIDER	0	0	0.3	0.82	0.82	0.54	0.5	0.87
EXTENDER	25	27.9	27.5	36.1	35.9	37.2	34.6	23.1
TOTAL	84	73	85.7	77	76.6	76.4	74.4	72.1

Results: FLOSS-AR

CANOPY	0	0	0	0	0	0	0	0
RECORD	0	0	0	0	0	0	0	0
DATAMANAGER	0.35	0.35	0	0	0	0	0	0
MP	1.0	1.2	1.4	1.6	1.6.2	1.8	1.8.2	2.0
SINK	15.3	3.9	15.6	4.14	3.5	2.7	2.78	2.45
OUTLINE	0	0	0	0	0	0	1.0	0.35
TRAIT	0	0	0	0	0	0	1.3	1.1
STATEMACHINE	0.71	0	0.66	0.82	0.82	0.54	0.5	5.4
PURETYPE	0	0	0	0	0.5	0.8	0.3	0.2
AUGMENTEDTYPE	0	0	0	0	0	0	0	0
PSEUDOCLASS	0	0	0	0	0	0	0	0
IMPLEMENTOR	0	0.71	0.3	0.27	0.27	0.27	0.25	0.35
OVERRIDER	0	0	0.3	0.82	0.82	0.54	0.5	0.87
EXTENDER	25	27.9	27.5	36.1	35.9	37.2	34.6	23.1
TOTAL	84	73	85.7	77	76.6	76.4	74.4	72.1

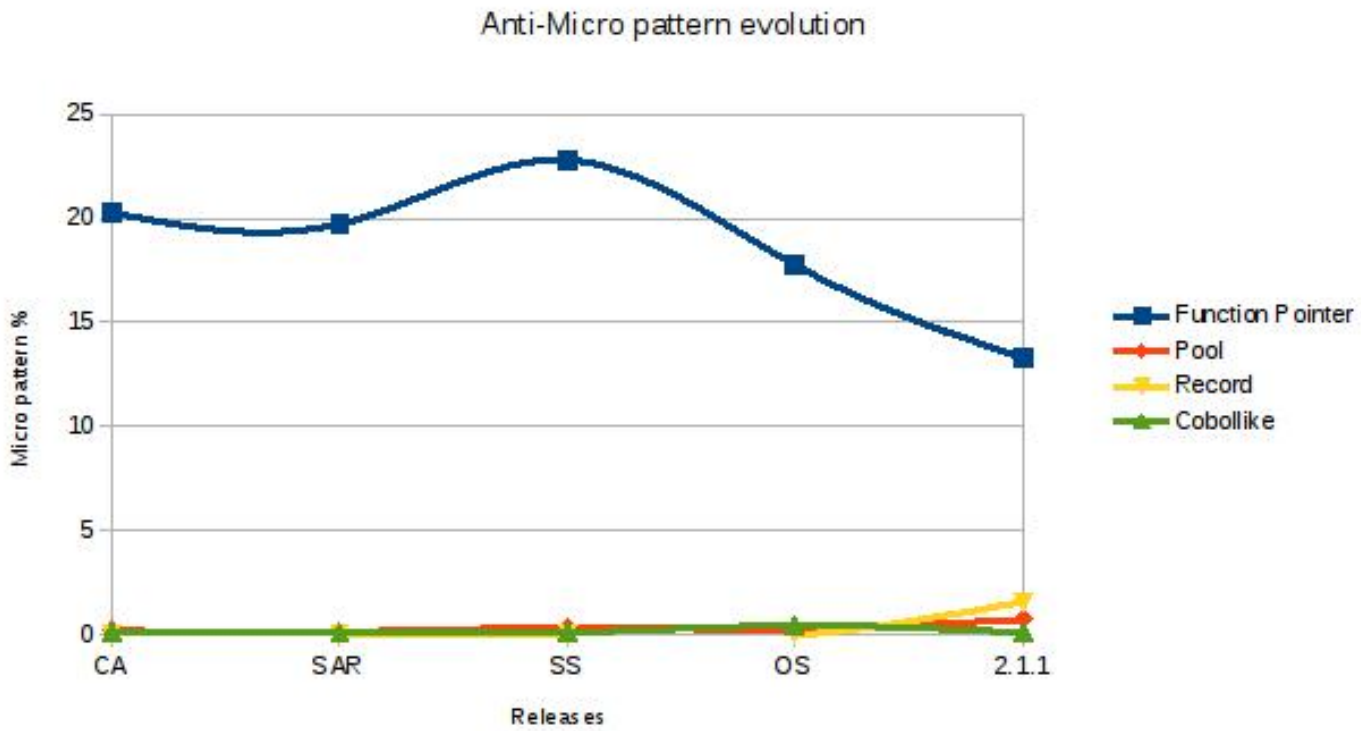
Gil-Maman rule is respected

Results: FLOSS-AR Fault Prone Micro pattern



FLOSS-AR fault-prone-micro pattern distribution evolution

Results: FLOSS-AR Anti-Micro pattern



FLOSS-AR anti-micro pattern distribution evolution

Results: JAPS

MP	CA	SAR	SS	OS	2.1.1
DESIGNATOR	1.5	1.5	1.6	1.38	0.9
TAXONOMY	0	0	0	0	0
POOL	0.2	0.2	0.36	0.3	0.76
JOINER	0	0	0	0	0
FUNCTIONPOINTER	20.2	19.7	22.8	17.8	13.31
FUNCTIONOBJECT	2.5	2.4	2	4.45	1.53
COBOLLIKE	0.17	0.17	0.14	0.46	0.13
STATELESS	0.4	0.3	0.29	1.07	2.57
COMMONSTATE	0.2	0.2	0.14	0.15	0.06
IMMUTABLE	0.2	0.2	0.14	0.76	0.06
RESTRICTEDCREATION	0.1	0.1	0.29	0.30	0.06
SAMPLER	0	0	0	0	0
BOX	2	2	3.21	0.15	13.79
COMPOUNDBOX	7.9	8.2	7.45	10.4	12.61

Results: JAPS

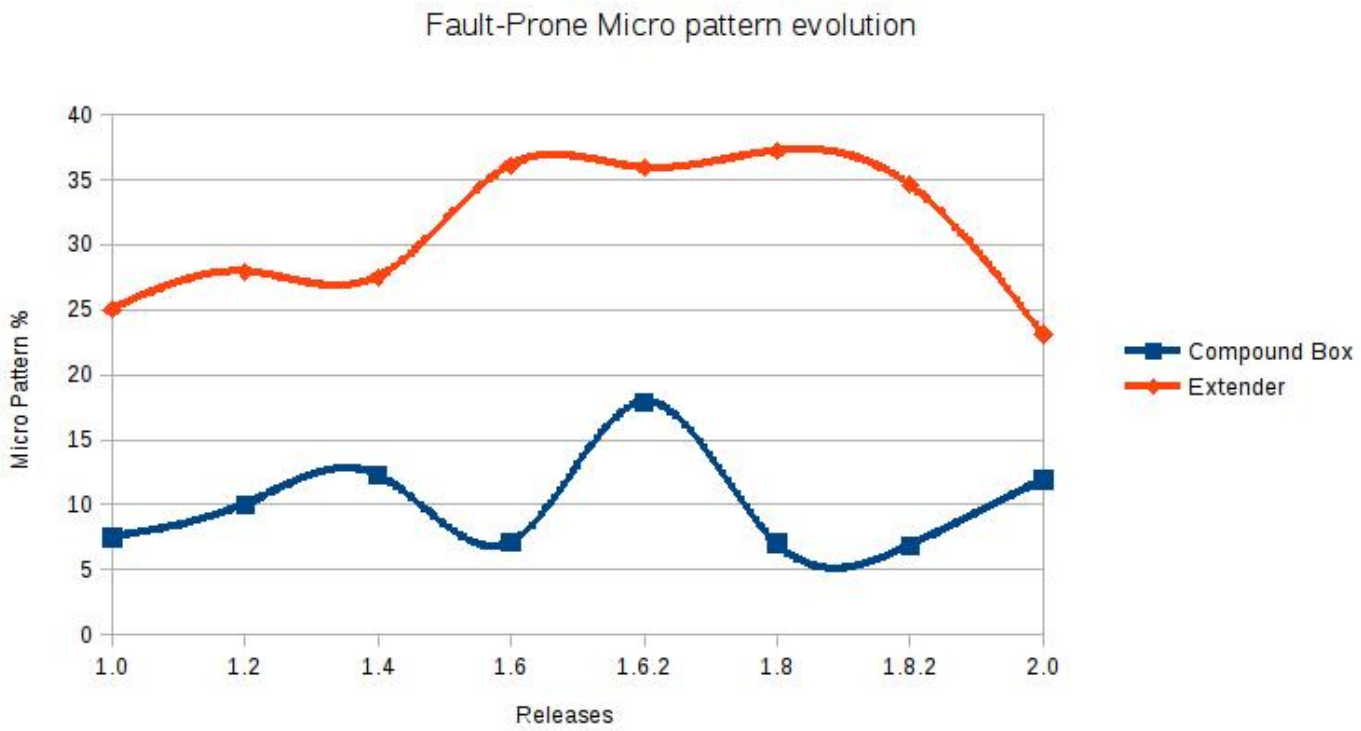
CANOPY	0	0	0	0	0
RECORD	0	0.2	0	0.2	1.6
DATAMANAGER	0	0	0	1.68	1.74
SINK	18.9	18.6	17.2	3.53	14.77
OUTLINE	0	0	0	0.3	1.1
TRAIT	0.33	0.3	0.29	1.2	0.13
STATEMACHINE	0.17	0.17	0.29	0.15	0.06
PURETYPE	0	0	0	0.3	0.1
AUGMENTEDTYPE	0	0	0	0	0
PSEUDOCLASS	0	0	0	0	0
IMPLEMENTOR	1.7	1.22	1.46	2.61	0.69
OVERRIDER	0.33	0.34	0.29	1.07	0.2
EXTENDER	28.4	28.8	27.7	28.4	16.58
TOTAL	85.1	84.8	85.8	75.5	81.6

Results: JAPS

CANOPY	0	0	0	0	0
RECORD	0	0.2	0	0.2	1.6
DATAMANAGER	0	0	0	1.68	1.74
SINK	18.9	18.6	17.2	3.53	14.77
OUTLINE	0	0	0	0.3	1.1
TRAIT	0.33	0.3	0.29	1.2	0.13
STATEMACHINE	0.17	0.17	0.29	0.15	0.06
PURETYPE	0	0	0	0.3	0.1
AUGMENTEDTYPE	0	0	0	0	0
PSEUDOCLASS	0	0	0	0	0
IMPLEMENTOR	1.7	1.22	1.46	2.61	0.69
OVERRIDER	0.33	0.34	0.29	1.07	0.2
EXTENDER	28.4	28.8	27.7	28.4	16.58
TOTAL	85.1	84.8	85.8	75.5	81.6

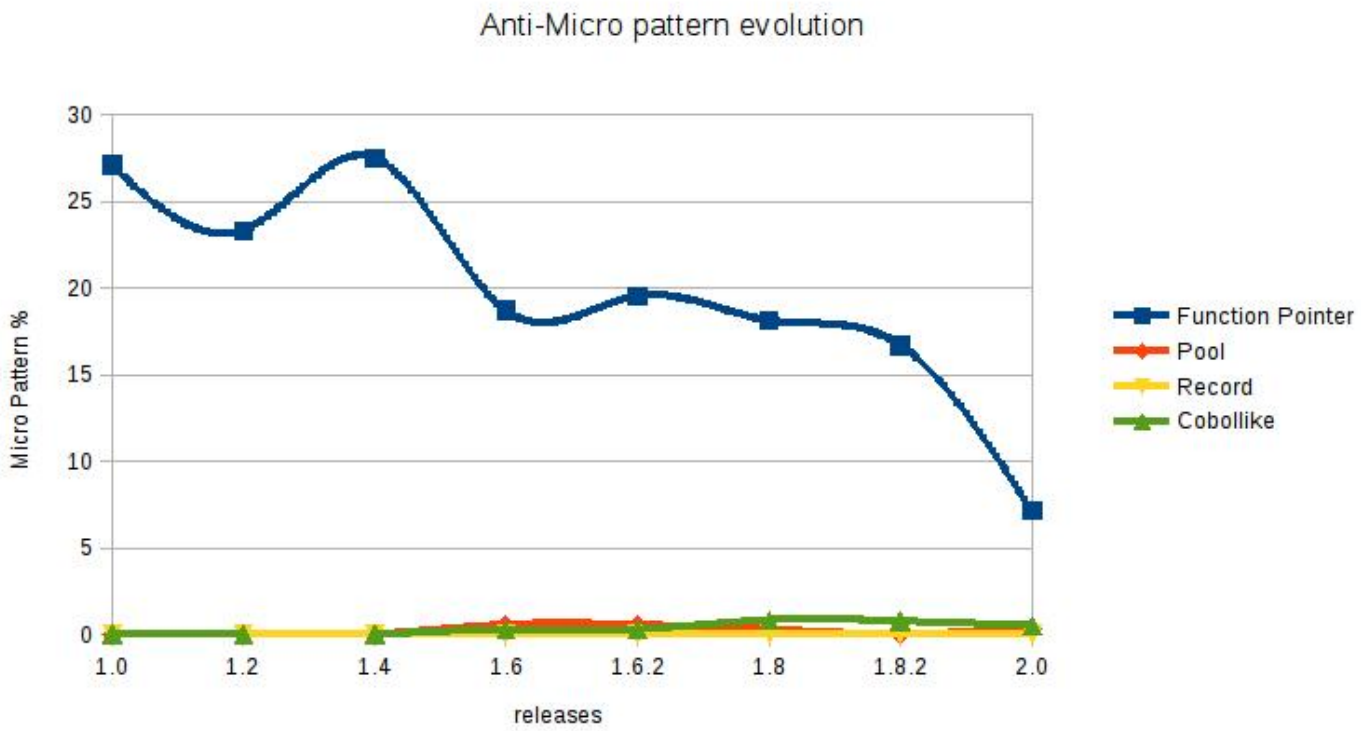
Gil-Maman rule is respected

Results: JAPS Fault Prone Micro pattern



JAPS fault-prone-micro pattern distribution evolution

Results: JAPS Anti-Micro pattern



JAPS anti-micro pattern distribution evolution

Fault analysis results: FLOSS-AR

		OS(%)	CA(%)	SAR(%)	SS(%)	2.1.1(%)
Distribution of faulty classes	NMP	63.12	62.41	71.63	70.92	23.4
	MP	36.87	37.58	28.36	29.07	76.59
Percentage of MP faults						
Fault Percentage of AMP		12.76	12.05	7.8	7.8	23.4
Fault Percentage of fault-prone MP faults		18.43	14.89	11.34	13.47	32.62
Fault Percentage of other MP		5.67	10.63	9.21	7.8	20.56

Fault analysis results: FLOSS-AR

		OS(%)	CA(%)	SAR(%)	SS(%)	2.1.1(%)
Distribution of faulty classes	NMP	63.12	62.41	71.63	70.92	23.4
	MP	36.87	37.58	28.36	29.07	76.59
Percentage of MP faults						
Fault Percentage of AMP		12.76	12.05	7.8	7.8	23.4
Fault Percentage of fault-prone MP faults		18.43	14.89	11.34	13.47	32.62
Fault Percentage of other MP		5.67	10.63	9.21	7.8	20.56

Non micro pattern are the most faulty micro pattern

Fault analysis results: FLOSS-AR

		OS(%)	CA(%)	SAR(%)	SS(%)	2.1.1(%)
Distribution of faulty classes	NMP	63.12	62.41	71.63	70.92	23.4
	MP	36.87	37.58	28.36	29.07	76.59
Percentage of MP faults						
Fault Percentage of AMP		12.76	12.05	7.8	7.8	23.4
Fault Percentage of fault-prone MP faults		18.43	14.89	11.34	13.47	32.62
Fault Percentage of other MP		5.67	10.63	9.21	7.8	20.56

Non micro pattern are the most faulty micro pattern

Among micro pattern anti-micro pattern and fault-prone micro patter are the most faulty

Conclusions and Future Works

- Our results show that agile practices decreased the distribution of fault-prone micro pattern and anti-micro pattern during software evolution, then such practices are able to increase the quality of the system.

Conclusions and Future Works

- Our results show that agile practices decreased the distribution of fault-prone micro pattern and anti-micro pattern during software evolution, then such practices are able to increase the quality of the system.
- We intend to collect more industrial project in order to verify and to extend the results of our research.

Questions