Testing Aspect-Oriented Software: Evolution and Collaboration through the Years

Fabiano Cutigi Ferrari, Erika Nina Höhn, José Carlos Maldonado
Departamento de Sistemas de Computação
Universidade de São Paulo - ICMC/USP
São Carlos, SP, Brazil
Email: {ferrari,hohn,jcmaldon}@icmc.usp.br

***Abstract*—Context: Research on testing aspect-oriented (AO) software has resulted in a variety of approaches derived from traditional testing techniques. However, there is neither a clear map of how they have evolved nor how researchers have collaborated so far. Objectives: To draw a general picture of research on testing AO software, focusing on the evolution of approaches and collaborations among researchers. Method: Our results rely on a systematic literature review that has been continually updated during the last years. Results: We identified a few testing approaches that have shown some evolution along the years. They are products of the largest collaboration groups, despite the lack of integration among groups. Conclusions: Although we noticed relative evolution in AO testing research, it still lacks more intensive collaboration in order to produce consolidated approaches that may become a common practice in AO software development.**

***Keywords*-Software testing; aspect-oriented software; systematic literature reviews; collaborative research; collaboration networks.**

I. Introduction

Aspect-Oriented Programming (AOP) [1] aims at improving the modularity and changeability of crosscutting concerns, which are typically spread across and tangled with multiple elements of a software system. As of the introduction of any new programming technique, AOP requires specific testing approaches in order to yield high quality products. In the last years, we can identify a number of research literature items focusing on aspect-oriented (AO) software testing (or simply *"AO testing"*). Research on this field comprises either: (i) the adaptation of traditional techniques and criteria, including underlying testing models such as control flow graphs and state models; or (ii) the proposition of new testing criteria to comply with particularities of AOP technologies, usually characterised by means of fault taxonomies and bug pattern catalogues.

Despite the availability of several approaches for AO testing, to date there is no clear map of their evolution since the early research presented by Zhao [2]. Moreover, collaboration among researchers in this field has yet not been characterised. Feedback on how AO testing has been evolving, added by an overview on how researchers have been working together, may contribute to the establishment of enhanced, well-founded testing approaches and collaboration networks that can leverage current, standalone research on AO testing. It can support the evolution of the community itself in the sense of stimulating or inducing "new connections" with different purposes (e.g. joint experiments and experience exchange). Having this in mind, in this paper we draw a general picture of research on testing AO software. We tackle the following questions:

- How have AO testing approaches been evolving along the years?
- How active have the researchers been? How much have they collaborated?

We are interested in approaches that focus on at least one of the traditional testing techniques, i.e. functional, structural, model or fault-based testing. Our results rely on the outcomes of a systematic literature review (SLR) that we have been continuously updating in the last years [3], [4], [5]. They show that structural-based testing has been the most investigated technique for AO software. Moreover, while some researchers have produced novel but standalone[1] contributions, others have been actively evolving their approaches towards desired levels of maturity. Results also show that the larger the collaboration group, the more evolved its contribution.

The remainder of this paper is organised as follows: Section II presents our data collection procedures and results, which includes a brief overview of our SLR planning and conduction. Section III brings our analysis and discussion about the SLR results in terms of evolution of AO testing and collaboration among researchers. Related work is summarised in Section IV. To conclude, Section V presents our final remarks and future work.

II. Data collection

This section summarises the data we collected during our SLR. Initially, we present our research goals and the steps

[1]We consider *standalone* some research that has not shown evolution since it has first been published.

we performed to identify all research of interest (Section II-A). In the sequence, we present the achieved results (Section II-B).

A. Planning and conducting the review

A systematic literature review is a rigorous, well-defined approach for identifying, evaluating and interpreting all available research with respect to (w.r.t.) a particular topic of interest [6]. The pieces of work that provide research evidences on a specific subject are called *primary studies*[2]. Our SLR on AO testing has two main goals: (i) identifying testing approaches that researchers and practitioners have been investigating for AO software; and (ii) identifying fault types that are specific to AO software. We defined the following primary research question (PQ1) and secondary questions (SQ1, SQ2 and SQ3), which one with associated inclusion and exclusion criteria:

- PQ1: *Which testing techniques/criteria have been applied to AO software to date?.*
- SQ1: *Which of these techniques/criteria are specific to AO software?*
- SQ2: *Which AO-specific fault types have been described to date?*
- SQ3: *Which kinds of experimental studies have been performed in order to validate AO software testing approaches?*

Defining an adequate search string is crucial for a SRL to succeed. It heavily relies on the expertise of the involved researchers and should be as comprehensive as possible in order to match all primary studies of interest. According to our research interests, we defined the following string[3].

```
(aspect-oriented software OR aspect-oriented application
OR aspect-oriented app OR aspect-oriented program OR aop)
AND  (testing OR fault OR defect OR error OR incorrect OR
failure)
```

The selected sources of primary studies range from indexed repositories (IEEE Xplore, ACM Digital Library, ScienceDirect and SpringerLink) to general purpose search engines (Google and Scirus). The general review procedures (i.e. preliminary selection, final selection, data extraction and review documentation) follow the Biolchini et al.'s template [8] and a process for multi-iteration SLRs [5]. In short, the preliminary selection consists of one going through specific parts of the primary studies to check whether they should be selected for full reading. In the final selection step, the researcher fully reads the paper and makes the final decision of selecting or discarding it, according to the inclusion and exclusion criteria. Data of interest is then extracted and stored in customised forms. During the whole process, documentation tasks are undertaken in order to enable auditability and replicability, which are two basic requirements for SLRs [6], [8].

[2]The term "primary study" has so far been used in the Evidence-based Software Engineering domain [7] to describe a variety of research results, from well-founded experimental procedures to incipient research approaches. Systematic reviews, on the other hand, are considered "secondary studies".

[3]This is the up-to-date version of the string without overlap of terms.

B. Results

Table I lists all primary studies selected during four iterations[4] performed over the past 36 months, with a medium interval of 12 months. Studies are sorted by year of publication, testing technique and authors. From hereafter, the notation *S#X* will be used to refer to the items listed in Table I, where X represents a study number listed in the first column of the table. Notice that most these studies are not listed in the References Section due to space limitations. However, we believe the information provided by the table (i.e. title, authors and publication year) helps the reader correctly identify each of them.

Table II summarises our SLR in terms of numbers of selected studies per testing technique. In studies where more than one testing technique is on the focus (i.e. studies that present hybrid approaches), each technique was proportionally taken into account. For example, considering study *S#5*, half point was awarded to structural-based and half point to state models-based. We made the decision of splitting points in order to keep the total number of selected studies consistent with Table I. Studies classified as "*other*" are of interest for at least one of the research questions, mostly SQ2.

Figure 1 depicts the distribution shown in Table II. On the left-hand side, we can see the how the number of studies varied year by year. The chart on the right-hand side shows the representativeness of each testing technique w.r.t. the total of selected studies. These numbers are analysed in the next section.

III. Analysis and discussion

This section brings our analysis and discussion based on the results of our SLR. We focus on two main issues: (i) the evolution of testing approaches; and (ii) the collaboration among researchers. A preliminary analysis of the data presented in Section II shows that, since 2004, there have not been great variations in research effort considering all testing techniques, on average. Regarding individual techniques, we can see that research on structural-based testing has resulted in the largest number of publications (17.5 out of 38 selected studies, i.e. 46% of the total).

We next try to answer the two questions presented in Section I. Notice that we only focus on the evolution of AO testing approaches. Therefore, studies classified as "*other*" in tables I and II (i.e. *S#3*, *S#4*, *S#10*, *S#11*, *S#12*, *S#22*, *S#23*, *S#24*, *S#31*, *S#32*, *S#40*, *S#41* and *S#50*) are not taken into account.

[4]The first iteration (also called "original review") has already been documented [3], [4].

Table I
SELECTED STUDIES.

Ref	Title	Authors	Technique	Year
1	Tool support for unit testing of aspect-oriented software	Zhao, J.	structural-based	2002
2	Data-flow-based unit testing of aspect-oriented programs	Zhao, J.	structural-based	2003
3	Towards the systematic testing of aspect-oriented programs	Alexander, R. T.; Bieman, J. M.; Andrews, A. A.	other	2004
4	Towards a practical approach to test aspect-oriented software	Zhou, Y.; Richardson, D. J.; Ziv, H.	other	2004
5	A state-based approach to testing aspect-oriented programs	Xu, D.; Xu, W.; Nygard, K.	structural and state models-based	2004
6	Adequate testing of aspect-oriented programs	Mortensen, M.; Alexander, R. T.	structural and fault-based	2004
7	Data flow integration testing criteria for aspect-oriented programs	Lemos, O. A. L.; Maldonado, J. C.; Masiero, P. C.	structural-based	2004
8	Unit testing of aspect-oriented programs	Lemos, O. A. L.; Vincenzi, A. M. R.; Maldonado, J. C.; Masiero, P. C.	structural-based	2004
9	A systematic aspect-oriented refactoring and testing strategy, and its application to JHotdraw	van Deursen, A.; Marin, M.; Moonen, L.	functional and structural-based	2005
10	Is AOP code easier or harder to test than OOP code?	Ceccato, M.; Tonella, P.; Ricca, F.	other	2005
11	Unit test your aspects	Lesiecki, N.	other	2005
12	Distributing classes with woven concerns: an exploration of potential fault scenarios	McEachen, N.; Alexander, R. T.	other	2005
13	Generating unit test sequences for aspect-oriented programs: towards a formal approach using UML state diagrams	Badri, M.; Badri, L.; Bourque-Fortin, M.	state models-based	2005
14	A state-based approach to testing aspect-oriented programs	Xu, D.; Xu, W.; Nygard, K.	state models-based	2005
15	Automated test generation for AspectJ programs	Xie, T.; Zhao, J.; Marinov, D.; Notkin, D.	structural and state models-based	2005
16	An approach for adequate testing of AspectJ programs	Mortensen, M.; Alexander, R. T.	structural and fault-based	2005
17	Structural unit testing of AspectJ programs	Lemos, O. A. L.; Maldonado, J. C.; Masiero, P. C.	structural-based	2005
18	Generating aspects-classes integration testing sequences: a collaboration diagram based strategy	Massicotte, P.; Badri, M.; Badri, L.	UML models-based	2005
19	A model-based approach to test generation for aspect-oriented programs	Xu, W.; Xu, D.	UML models-based	2005
20	Automated testing of pointcuts in AspectJ programs	Anbalagan, P.	fault-based	2006
21	Efficient mutant generation for mutation testing of pointcuts in aspect-oriented programs	Anbalagan, P.; Xie, T.	fault-based	2006
22	A fault model for pointcuts and advice in AspectJ programs	Bækken, J. S.	other	2006
23	A candidate fault model for AspectJ pointcuts	Bækken, J. S.; Alexander, R. T.	other	2006
24	Towards a fault model for AspectJ programs: step 1 pointcut faults	Bækken, J. S.; Alexander, R. T.	other	2006
25	State-based incremental testing of aspect-oriented programs	Xu, D.; Xu, W.	state models-based	2006
26	State-based testing of integration aspects	Xu, W.; Xu, D.	state models-based	2006
27	A framework and tool supports for generating test inputs of AspectJ programs	Xie, T.; Zhao, J.	structural and state models-based	2006
28	Testing aspect-oriented programming pointcut descriptors	Lemos, O. A. L.; Ferrari, F. C.; Masiero, P. C.; Lopes, C. V.	structural and fault-based	2006
29	Testing during refactoring: adding aspects to legacy systems	Mortensen, M.; Ghosh, S.; Bieman, J. M.	structural-based	2006
30	Aspects-classes integration testing strategy: an incremental approach	Massicotte, P.; Badri, L.; Badri, M.	UML models-based	2006
31	Debugging aspect-enabled programs	Eaddy, M.; Aho, A.; Hu, W.; McDonald, P.; Burger, J.	other	2007
32	On identifying bug patterns in aspect-oriented programs	Zhang, S.; Zhao, J.	other	2007
33	Testing aspect oriented programs: an approach based on the coverage of the interactions among advices and methods	Bernardi, M.L.; Di Lucca, G.A.	structural-based	2007
34	Pairwise structural testing of object and aspect-oriented Java programs	Franchin, I. G.; Lemos, O. A. L.; Masiero, P. C.	structural-based	2007
35	Control and data flow structural testing criteria for aspect-oriented programs	Lemos, O. A. L.; Vincenzi, A. M. R.; Maldonado, J. C.; Masiero, P. C.	structural-based	2007
36	Towards a tool supporting integration testing of aspect-oriented programs	Massicotte, P.; Badri, L.; Badri, M.	UML models-based	2007
37	Generation of test requirements from aspectual use cases	Xu, D.; He, X.	UML models-based	2007
38	Automated generation of pointcut mutants for testing pointcuts in AspectJ programs	Anbalagan, P.; Xie, T.	fault-based	2008
39	Mutation testing for aspect-oriented programs	Ferrari, F.C.; Maldonado, J.C.; Rashid, A.	fault-based	2008
40	Assessing the impact of aspects on exception flows: an exploratory study	Coelho, R.; Rashid, A.; Garcia, A.; Ferrari, F.; Cacho, N.; Kulesza, U.; von Staa, A.; Lucena, C.	other	2008
41	Exception handling bug patterns in aspect oriented programs	Coelho, R.; Rashid, A.; Kulesza, U.; von Staa, A.; Lucena, C.; Noble, J.	other	2008
42	A state-based testing approach for aspect-oriented programming	Liu, C-H.; Chang, C-W.	state models-based	2008
43	Reverse engineering of aspect oriented systems to support their comprehension, evolution, testing and assessment	Bernardi, M.L.	structural-based	2008
44	Testing coverage criteria for aspect-oriented program	Bernardi, M.L.; Di Lucca, G.A.	structural-based	2008
45	Using structural testing to identify unintended join points selected by pointcuts in aspect-oriented programs	Lemos, O. A. L.; Masiero, P. C.	structural-based	2008
46	Integration testing of aspect-oriented programs: a structural pointcut-based approach	Lemos, O. A. L.; Masiero, P. C.	structural-based	2008
47	Testing aspect-oriented programs with UML design models	Xu, D; Xu, W.; Wong, W. E.	UML models-based	2008
48	A test-driven approach to developing pointcut descriptors in AspectJ	Delamare, R.; Baudry, B.; Ghosh, S.; Le Traon, Y.	functional and fault-base	2009
49	Fault model and test-case generation for the composition of aspects	Babu, C.; Krishnan, H. R.	UML models-based	2009
50	Enabling the adoption of aspects - testing aspects: a risk model, fault model and patterns	Kumar, N.; Sosale, D.; Konuganti, S. N.; Rathi, A.	other	2009
51	Integration testing of object-oriented and aspect-oriented programs: a structural pairwise approach for Java	Lemos, O. A. L.; Franchin, I. G.; Masiero, P. C.	structural-based	2009

Table II
DISTRIBUTION OF SELECTED STUDIES ALONG THE YEARS.

	2002	2003	2004	2005	2006	2007	2008	2009	Total	Average	Average 2004 - 2009*
Structural-based	1	1	3	2.5	2	3	4	1	**17.5**	**2.2**	**2.8**
State models-based	0	0	0.5	2.5	2.5	0	1	0	**6.5**	**0.8**	**1.2**
Fault-based	0	0	0.5	0.5	2.5	0	2	0.5	**6**	**0.8**	**1.1**
UML models-based	0	0	0	2	1	2	1	1	**7**	**0.9**	**1.3**
Functional-based	0	0	0	0.5	0	0	0	0.5	**1**	**0.1**	**0.2**
Other	0	0	2	3	3	2	2	1	**13**	**1.6**	**2.4**
Total	**1**	**1**	**6**	**11**	**11**	**7**	**10**	**4**	**51**	**6.4**	**8.9**
Average	**0.2**	**0.2**	**1.0**	**1.8**	**1.8**	**1.2**	**1.7**	**0.7**	**8.5**		

* considering only the 1st. semester of 2009

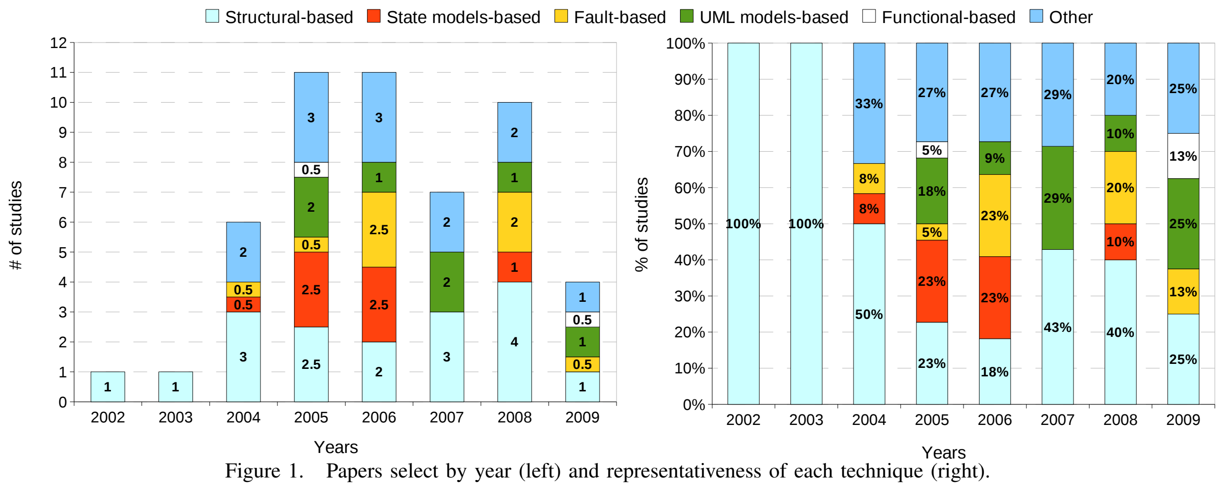

Figure 1. Papers select by year (left) and representativeness of each technique (right).

A. *How have AO testing approaches been evolving along the years?*

Structural-based testing: : The first initiatives on structural-based testing by Zhao (*S#1* and *S#2*) proposed control and data flow models for the current implementation of the AspectJ language. Although Zhao's research has not evolved ever since, the figures presented in Table II show that structural-based testing has been the most investigated technique so far. We can highlight Lemos et al.'s and Franchin et al.'s contribution as the most mature (*S#7*, *S#8*, *S#17*, *S#28*, *S#34*, *S#35*, *S#45*, *S#46* and *S#51*) in terms of formally defined testing models (control and data flow models), associated criteria and tooling support. Other purely structural-based contributions either has not shown evolution in the last years (*S#29*) or are still incipient (*S#33*, *S#43*, *S#44*). Moreover, considering hybrid approaches that include structural elements of the code for deriving test requirements (*S#5*, *S#6*, *S#9*, *S#15*, *S#16* and *S#27*), in general they tended to focus more on the other techniques and will be discussed in the sequence.

State models-based testing: : When we consider research on state models-based AO testing, we can identify two main contribution lines and standalone (*S#13*) and incipient contributions (*S#42*). Xu et al.'s approach (*S#5*, *S#14*, *S#25* and *S#26*) has shown some evolution from 2004 to 2006. It comprises the definition of state models that take into account aspect-related elements (e.g. advices that interfere in the system state), from which state transition trees and test cases are derived. The second research line (*S#15* and *S#27*) comprises automated test case generation for AO programs based on possible states of the system and internal structural elements. Both approaches, however, seems to be stagnated since the last papers were published in 2006.

Fault-based testing: : It has shown gradual evolution through the years. The research of Mortensen and Alexander (*S#6* and *S#16*) was published in 2004 and 2005 and includes high-level definitions of mutation operators for AO programs. Anbalagan and Xie (*S#20*, *S#21* and *S#38*) provided real implementation of those operators, focusing on a subset of AspectJ elements, the pointcut expressions. Ferrari et al.

(*S#39*), based on an extensive analysis of AO technologies and fault types, designed a set of mutation operators for AspectJ programs that subsumes all previously proposed operators. Part of these mutation operators were recently implemented by Delamare et al. (*S#48*) to compose a hybrid testing approach that includes test-driven development and mutation testing.

UML models-based testing: : As well as for state models-based testing, we can identify two main lines for UML models-based testing, and also standalone (*S#37*) and incipient contributions (*S#49*). The more recent approach is presented by Xu et al. (*S#19* and *S#46*) and relies on extended class and sequence diagrams to derive required test sequences. Similar approach was proposed by Massicotte et al. (*S#18*, *S#30* and *S#36*), however based on class and collaboration diagrams.

Functional-based testing: : Generally, we identified too little research comprising functional testing (only *S#9* and *S#48*). Furthermore, this technique was not the focus of the selected studies. Notice that we did not consider research on regression testing for AO software, which usually includes tasks related to functional-based testing (e.g. double-checking just modified system functionalities).

B. How active have the researchers been? How much have they collaborated?

Analysis of Table I shows that a total of 63 researchers are involved in at least one selected study. A closer look at the figures reveals that since 2004, researchers like R. T. Alexander, O. A. L. Lemos and D. Xu have been publishing at least one primary study per year, on average. On the other hand, we can see that 16 researchers have published a single piece of work regarding AO testing techniques and fault models up to 2006, but have not published ever since. This lack of continuation suggests that such researchers have not kept active in the AO testing field. We call the reader's attention to the fact that we are not evaluating the quality or the impact of a study. For example, Zhao's work [2], [9] is probably the most cited among all selected studies, despite the lack of evolution of the original contribution. In fact, a contribution may have great impact in future research even though it might have not evolved since it has been originally published.

Figure 2 shows the map of collaboration among researchers. Analysing each collaboration group individually, four of the largest (labels 6, 9, 12 and 14 in Figure 2) are responsible for the approaches that have shown the highest maturity levels, as discussed in Section III-A. On the other hand, many authors that are involved in small collaboration groups (up to 3 researchers) have published either standalone contributions (e.g. *S#4*, *S#9*, *S#10* and *S#11*) or still incipient contributions (e.g. *S#42*, *S#49*).

The collaboration network structure has important implications for the spread of information. According to Newman [10], for example, it is clear that any variation in the average number of acquaintances that individuals have might substantially influence the propagation of information. Therefore, the map presented in Figure 2 can be used as a way of encouraging future collaborations among groups. For example, in the previous section, we saw that fault-based AO testing has gradually evolved during the last years. However, a further look at Figure 2 reveals that three disconnected groups (6, 9 and 12) worked separately in such evolving approach. Another example regards structural-based testing, where groups 6 and 5 have been working on similar approaches (i.e. definition of inter-procedural control flow graphs and associated testing criteria) although no collaborations have been established.

IV. Related work

This work extends our original SLR results [3], [4] in two ways: (i) updating the state-of-the-art in AO testing research, focusing on techniques and associated criteria; and (ii) analysing its evolution and the collaboration of involved researchers through the time. Amar and Shabbir [11] also present a SLR on AO testing. However, they focus on two main subjects: (i) challenges, issues and problems for AO testing; and (ii) structural-based testing approaches for AO software. Their search included primary studies published up to May 2008. Despite the differences among goals, there is an overlap between the set of studies selected in our SLR and Amar and Shabbir's. Considering the list presented in Table I and the 43 studies selected by them, 18 are of interest for both surveys (*S#2*, *S#3 S#9*, *S#12*, *S#13*, *S#14*, *S#18*, *S#21*, *S#24 S#25*, *S#26*, *S#27*, *S#28*, *S#30*, *S#31*, *S#32*, *S#35* and *S#33*). However, Amar and Shabbir do not evaluate the evolution of AO testing approaches neither the collaboration of researchers in this field.

We can also identify a few surveys and evaluations of AO testing approaches [12], [13], [14]. Naqvi et al. [12] evaluate three approaches for AO testing w.r.t. their ability to reveal the fault types characterised by Alexander et al. [15]. Parizi and Ghani [13] extend Naqvi et al.'s work, including two other approaches for AO testing. They also highlight some gaps of each piece of research (e.g. the weaknesses and required enhancements/improvements). However, authors of both studies do not make clear how they have selected such approaches. Moreover, evolution is not taken into consideration. Xie and Zhao[14] present their AO testing perspectives based on test input generation and coverage measurement of AO programs. Again, the scope is limited to a few approaches, mostly based on their own research.

Regarding collaborative research, we could not find any analysis of collaboration networks specific for AO testing, although there are studies that examine the scientific collaboration through authorship networks. An example is the

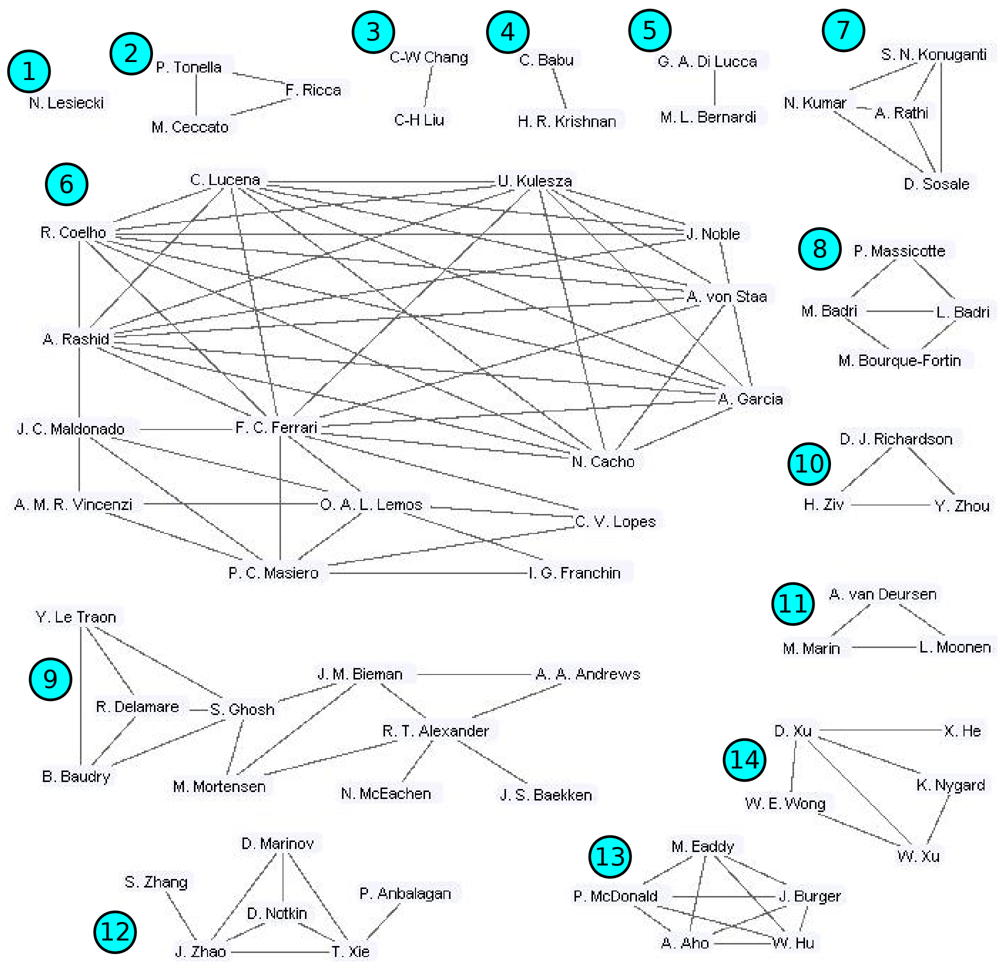

Figure 2. Researchers' collaborations networks.

work of Hassan and Holt [16], who present an analysis of research in software reverse engineering. According to them, publications in a research community give a picture of the progress of collaboration and emergence of topics in an active research field.

V. Final remarks and future work

This paper presented the results of a multi-iteration systematic literature review on AO software testing. The results were analysed from two points of view: (i) how AO testing has evolved since the early structural-based approach presented by Zhao [2]; and (ii) the collaboration among people who have carried out research on AO testing.

We identified a few testing approaches that have shown evolution along the years. Structural and fault-based testing seem to have reached the highest maturity levels when compared to the other techniques, including the definition of underlying testing models and adequate tooling support for the associated test selection criteria. Both models and tools have evolved in a sequence of refinements and enhancements.

However, we can still notice a lack of collaboration among research groups, in order to produce consolidated approaches that may become a common practice in AO software development. It is even more motivated by the fact that the most mature research is product of the largest research groups, in which collaboration was more noticeable. Joint research should target, for example, the evaluation of testing approaches within different application domains, either by means of exploratory studies or controlled experiments. This requires effective sharing of knowledge and infrastructure (e.g. underlying literature, tools and benchmarks) and would result in more mature approaches to be applied in practice.

Our future research includes the full documentation of the SLR. We will address every SLR research question as defined in Section II-A. Documenting the SLR will also include: (i) defining quality criteria to be applied to the selected studies; and (ii) providing details of each testing approach, including the underlying techniques and associated test selection criteria. The quality criteria should take into account, for example, the quality of the journal/conference a study was published in, and how the proposed testing approach has been assessed. Regarding the research collaboration network, further analysis of its historical evolution may increase the interaction within the community, for example, through the promotion of AO testing-related events. We also consider the creation of a knowledge portal for AO testing, which should include regular updates of the systematic review.

Acknowledgment

This work was financially supported by FAPESP/Brazil (Process 05/55403-6), CAPES/Brazil (Process 0653/07-1) and CNPq/Brazil (Process 552875/2008-2).

References

[1] G. Kiczales, J. Irwin, J. Lamping, J.-M. Loingtier, C. Lopes, C. Maeda, and A. Menhdhekar, "Aspect-oriented programming," in *ECOOP'1997*. Springer-Verlag, 1997, pp. 220–242 (LNCS v.1241).

[2] J. Zhao, "Tool support for unit testing of aspect-oriented software," in *Workshop on Tools for Aspect-Oriented Software Development*, Seattle/WA - USA, 2002.

[3] F. C. Ferrari and J. C. Maldonado, "A systematic review on aspect-oriented software testing," in *WASP'2006*, Florianópolis/SC - Brasil, 2006, pp. 101–110, (in Portuguese).

[4] ——, "Aspect-oriented software testing: A systematic review," ICMC/USP, São Carlos - Brasil, Tech. Report 291, 2007, (in Portuguese).

[5] ——, "Experimenting with a multi-iteration systematic review in software engineering," in *ESELAW'2008*, Salvador/BA - Brazil, 2008.

[6] B. Kitchenham, "Procedures for performing systematic reviews," Department of Computer Science - Keele University and National ICT Australia Ltd, Keele/Staffs-UK and Eversleigh-Australia, Joint Tech. Report TR/SE-0401, 2004.

[7] B. A. Kitchenham, T. Dybå, and M. Jørgensen, "Evidence-based software engineering," in *ICSE'2004*. Edinburgh - Scotland: IEEE Computer Society, 2004, pp. 273–281.

[8] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, "Systematic review in software engineering," Systems Engineering and Computer Science Dept., COPPE/UFRJ, Rio de Janeiro - Brazil, Tech. Report RT-ES 679/05, 2005.

[9] J. Zhao, "Data-flow-based unit testing of aspect-oriented programs," in *COMPSAC'2003*. IEEE Computer Society, 2003, pp. 188–197.

[10] M. E. J. Newman, "The structure of scientific collaboration networks," *Proceedings of the National Academy of Sciences of the USA*, vol. 98, no. 2, pp. 404–409, 2001. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.021544898

[11] M. Amar and K. Shabbir, "Systematic review on testing aspect-oriented programs," Master's thesis, School of Engineering, Blekinge Institute of Technology, Ronneby - Sweden, 2008. [Online]. Available: http://www.bth.se/fou/cuppsats.nsf/1d345136c12b9a52c1256608004f0519/a2dcfd1f088383abc12574cd004b23c8!OpenDocument

[12] S. A. A. Naqvi, S. Ali, and M. U. Khan, "An evaluation of aspect oriented testing techniques," in *Proceedings of the IEEE Symposium on Emerging Technologies*. Islamabad - Pakistan: ACM Press, 2006, pp. 461–466.

[13] R. M. Parizi and A. A. Ghani, "A survey on aspect-oriented testing approaches," in *ICCSA'2007*. Kuala Lumpur - Malaysia: IEEE Computer Society, 2007, pp. 78–85.

[14] T. Xie and J. Zhao, "Perspectives on automated testing of aspect-oriented programs," in *WTAOP'2007*. Vancouver/British Columbia - Canada: ACM Press, 2007, pp. 7–12.

[15] R. T. Alexander, J. M. Bieman, and A. A. Andrews, "Towards the systematic testing of aspect-oriented programs," Dept. of Computer Science, Colorado State University, Fort Collins-USA, Report CS-04-105, 2004.

[16] A. E. Hassan and R. C. Holt, "The small world of software reverse engineering," in *WCRE 2004*. Delft - The Netherlands: IEEE Computer Society, 2004, pp. 278– 283. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1374327&isnumber=30025