ARE CISQ RELIABILITY MEASURES PRACTICAL? A RESEARCH PERSPECTIVE

Johannes Bräuer, Reinhold Plösch, Manuel Windhager

TAIC PART 2017, Tokyo

JOHANNES KEPLER
UNIVERSITÄT LINZ

WHAT TO EXPECT?

- Motivation for measuring reliability
- CISQ Standard
- Measuring tool MUSE
- Project of study HSQLDB
- Reliability analysis and findings
- Contributions
- Collaboration with industrial partner

MOTIVATION FOR MEASURING RELIABILITY

- Economic impact of reliability issues can get enormous when resource consumption, test cycles, feature deployment time, or maintenance costs increase
  - Some reliability measurement techniques relying on external data (time between failures, count failures in operation, etc.)
  - Black box techniques verify input and try to predict faults
- Static analysis identifies more issues in a shorter time [1]
- Static analysis approaches implemented as quality models make software quality (reliability) tangible
- Gap remains between the abstract quality characteristics in a quality model and concrete measurements

THE CISQ STANDARDS

Consortium for IT Software Quality (CISQ) tries to close this gap [2]

AUTOMATED SOURCE CODE RELIABILITY MEASURE (ASCRM)

- 29 software reliability measures defined in the standard, example:

	ASCRM-CWE-397: Declaration of Throws for Generic Exception
Consequence	Software without consistent and complete handling of errors and exceptions makes it impossible to accurately identify and adequately respond to unusual and unexpected situations
Objective	Avoid failure to use dedicated exception types
Measure Element	Number of instances where the named callable control element or method control element throws an exception parameter whose data type is part of a list of overly broad exception data types.

- Goal: Are the measures provided by CISQ practical for the automatic measurement of the quality attribute reliability?
  - ☐ *Are the specifications detailed enough?*
  - ☐ *Can a tool be provided to support automatic measurement?*

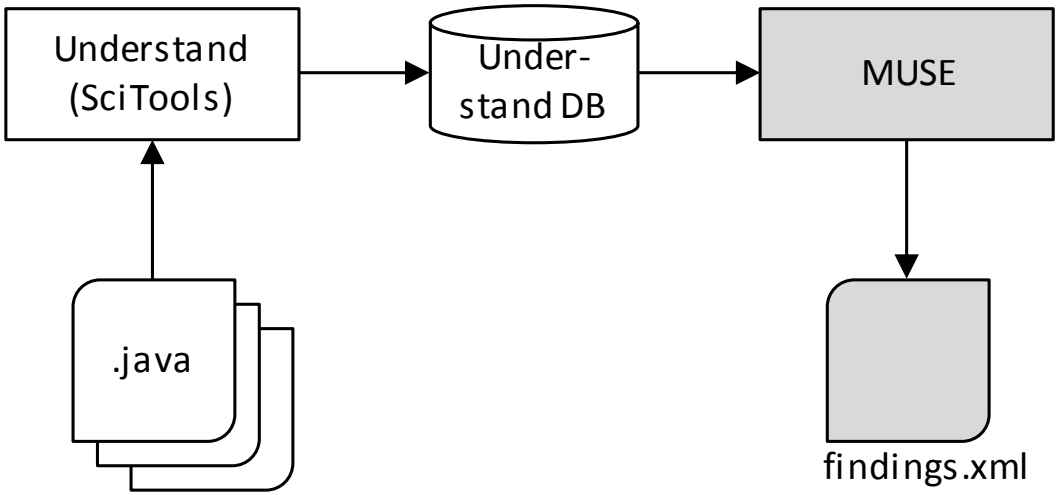
ANALYZING CISQ MEASURES FOR RELIABILITY

- Five measures are non-automatable, seven measures are unapplicable for Java → 17 remaining measures
- Assumptions are required for the interpretation of the specifications, example:

Reliability Pattern	ASCRM-CWE-397: Declaration of Throws for Generic Exception
Measure Element	Number of instances where the named callable control element or method control element throws an exception parameter whose data type is part of a list of overly broad exception data types.
Assumption	Since there is no reference to a list of broad exception data types we consider *java.lang.Exception* as the broadest data type and return a violation when it is used in a throw clause (CWE-397).

MUSE – A FRAMEWORK FOR MEASURING SOURCE CODE

- MUSE (Muse Understand Scripting Engine) [5]
- Parsing of source code done by Understand (SciTools)
- MUSE accesses provided information using the Perl API
- MUSE is a framework + tool for the 17 CISQ measures



PROJECT OF STUDY: HyperSQL

- HyperSQL DataBase (HSQLDB) leading relational SQL-database software system implemented in Java
- Development started in 2001 by HSQL Development Group
- Database engine for 1,700 open source and commercial products
- Source code is publicly available

Claim: „*HSQLDB is a mature product. The 2.3.x series was launched three years ago with enhanced reliability and performance compared to previous releases*.” (Source: http://hsqldb.org/)

EX-POST VIEW ON RELIABILITY ENHANCEMENT

- *Research interest*: Can the CISQ measures for reliability reflect the concentration on reliability during the development?
- Investigation of five HSQLDB versions:

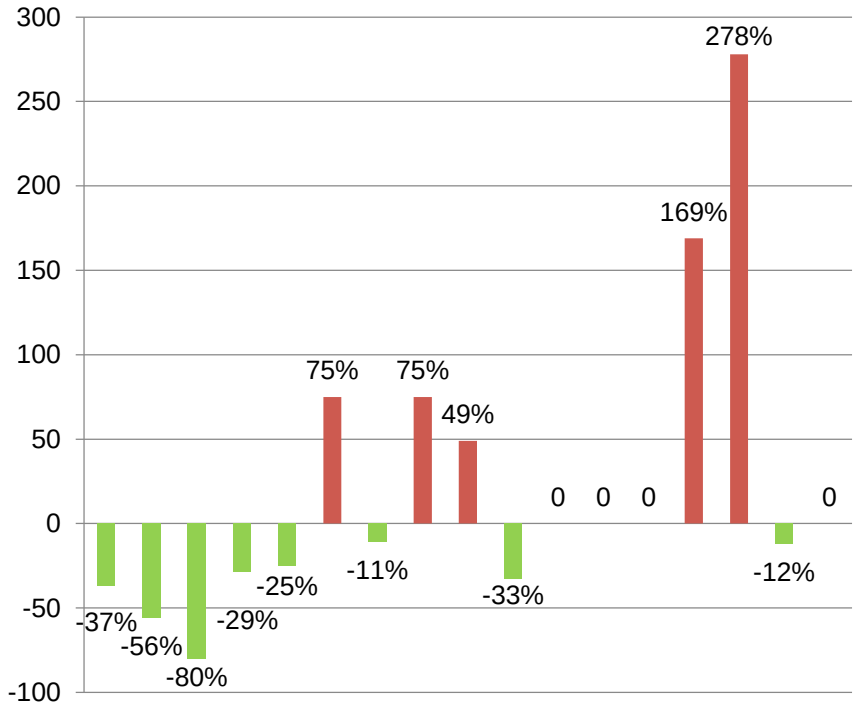
Version	Release Date	Lines of Code (LOC)	Classes
2.3.3	02.07.2015	143.101	538
2.2.0	11.05.2011	128.365	489
2.0.0	06.06.2010	128.064	534
1.8.0	02.06.2008	48.827	231
1.7.3	07.02.2005	41.844	202

- Number of rule violations are normalized with LOC

EX-POST VIEW ON RELIABILITY ENHANCEMENT

ID / Version	1.7.3	1.8.0	2.0.0	2.2.0	2.3.3	1.7.3 – 2.3.3
CWE-252-d.	0.17	0.16	0.08	0.11	0.10	-37%
CWE-396	2.96	2.56	1.54	1.36	1.30	-56%
CWE-397	0.96	0.72	0.27	0.19	0.20	-80%
CWE-674	0.55	0.39	0.35	0.37	0.39	-29%
RLB-1	0.50	0.49	0.40	0.44	0.38	-25%
RLB-2	0.02	0.02	0.05	0.03	0.04	+75%
RLB-3	0.00	0.00	0.02	0.02	0.02	-11%[a]
RLB-4	0.02	0.02	0.04	0.03	0.04	+75%
RLB-6	0.00	0.00	0.02	0.03	0.03	+49%[a]
RLB-8	0.00	0.00	0.06	0.05	0.04	-33%[a]
RLB-9	0.00	0.00	0.00	0.00	0.00	-
RLB-11	0.00	0.00	0.00	0.00	0.00	-
RLB-12	0.00	0.00	0.00	0.00	0.00	-
RLB-13	0.24	0.51	0.69	0.68	0.64	+169%
RLB-14	0.41	0.33	1.51	1.60	1.54	+278%
RLB-18	0.02	0.04	0.05	0.02	0.02	-12%
RLB-19	0.00	0.00	0.00	0.00	0.00	-
Findings / KLOC	5.86	5.24	5.08	4.93	4.75	-19%

[a] Version 2.0.0 is used as denominator for calculating the change in percentage.



IMPROVEMENT OF RELIABILITY

- *Research interest*: Which CISQ measures will be fixed based on the measurement result?

- Cooperation with HSQLDB development team
- Providing measuring result on SonarQube[1] and support for analyzing rule violations
- Comparison of last official release (2.3.3) and working branch in HSQLDB git repository

[1] https://www.sonarqube.org/

IMPROVEMENT OF RELIABILITY

ID / Version	2.3.3		trunk for 2.3.4		Improve-ments	
	abs.	norm.	abs.	norm.	abs.	norm.
CWE-252-d.	15	0.10	12	0.08	3	-20%
CWE-396	186	1.30	178	1.24	8	-5%
CWE-397	28	0.20	27	0.19	1	-4%
CWE-674	56	0.39	56	0.39	0	0%
RLB-1	54	0.38	51	0.36	3	-6%
RLB-2	6	0.04	2	0.01	4	-67%
RLB-3	3	0.02	0	0.00	3	-100%
RLB-4	6	0.04	4	0.03	2	-34%
RLB-6	5	0.03	5	0.03	0	0%
RLB-8	6	0.04	6	0.04	0	0%
RLB-9	0	0.00	0	0.00	-	-
RLB-11	0	0.00	0	0.00	-	-
RLB-12	0	0.00	0	0.00	-	-
RLB-13	92	0.64	92	0.64	0	0%
RLB-14	220	1.54	220	1.53	0	0%
RLB-18	3	0.02	3	0.02	0	0%
RLB-19	0	0.00	0	0.00	-	-
Total	680	4.752	656	4.573	24	-3.5%

- CWE-396: Developer team decided to be more specific with the handling of various exceptions
- RLB-3: Changed the data type of fields for supporting serialization
- RLB-2, RLB-4: Reconsidered the implementation of, e.g., the *java.io.Serializable* interface (over-engineered)

CONTRIBUTIONS

- Static testing tool that allows measuring 17 of the 29 CISQ reliability measures in Java
  - 12/29 are not applicable for Java
  - 11/17 require an assumption
- 4 CISQ measures could not identify any rule violations
- 7 CISQ measures identified important findings with high interest to get addressed
- 3 CISQ measures are considered as important and practical
- 2 CISQ measures are less important with respect to reliability
- 1 CISQ measures could identify just an indicator of reliability issues and would need tool support beyond static code analysis

CONCLUSION

- 14 of the 17 implemented CISQ measures are suitable for measuring reliability
- For the 12 unimplemented CISQ measures their suitability for measuring reliability cannot be stated

- Study did not reveal the degree of coverage the CISQ measures achieved for reliability
  - *Future work*:
    - Survey or systematic literature study to identify the uncovered white-spots of the CISQ standard
    - Collaboration with CISQ and proposing our implementations as mean for verifying their measures

COLLABORATION WITH INDUSTRIAL PARTNER

- How did you get in contact with the industrial partner?
  - Providing a concrete measuring result to project lead
- How did you collaborate with the industrial partner? (win-win approach)
  - Offering service of verifying HSQLDB based on the CISQ measures
  - Getting insights in the practicability of the measures
- How long have you collaborated with the industrial partner?
  - Over a time period of 2 month.
- What challenges/success factors did you experience when collaborating with the industrial partner?
  - Engineers are concerned about side-effects
  - Code ownership restricts the modification of software parts
  - Engineers need actionable reports of measuring results
  - Be concerned about false-positives since they disrupt

JOHANNES KEPLER
UNIVERSITÄT LINZ

THANK YOU!

JOHANNES BÄUER

Johannes Kepler University Linz
Department of Business Informatics – Software Engineering

johannes.braeuer@jku.at

JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
www.jku.at

REFERENCES

[1] R. Scandariato, J. Walden, and W. Joosen, “Static analysis versus penetration testing: A controlled experiment,” in *Proc. of 24th Int. Symposium on Software Reliability Engineering (ISSRE)*, 2013, pp. 451–460.

[2] R. M. Soley and B. Curtis, “The consortium for IT software quality (CISQ),” in *Software Quality. Increasing Value in Software and Systems Development*, D. Winkler, S. Biffl, and J. Bergsmann, Eds. Springer Berlin Heidelberg, 2013, pp. 3–9.

[3] J. Bräuer, R. Plösch, and M. Saft, “Measuring maintainability of OO-Software - Validating the IT-CISQ quality model,” in *Proc. of 2015 Federated Conference on Software Development and Object Technologies*, Zilina, Slovakia, 2015, pp. 283-301.

[4] R. Plösch, S. Schürz, and C. Körner, “On the validity of the IT-CISQ quality model for automatic measurement of maintainability,” *in Proc. of 39th Annu. Int. Computers, Software & Applications Conference (COMPSAC 2015)*, Taichung, Taiwan, 2015, pp. 326-334.

[5] R. Plösch, J. Bräuer, C. Körner, and M. Saft, “MUSE - Framework for measuring object-oriented design,” *J. Object Technol.*, vol. 15, no. 4, pp. 2:1-29, Aug. 2016.