Universiteit Leiden
The Netherlands

The effects of UML modeling on the quality of software

Nugroho, A.

Citation
Nugroho, A. (2010, October 21). *The effects of UML modeling on the quality of software*. Retrieved from https://hdl.handle.net/1887/16070

Version:	Corrected Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/16070

Note: To cite this publication please use the final published version (if applicable).

Appendix A

UML Survey Questionnaire

Generally, UML designs are the basis for an implementation of a system. For example, an implementation has the same classes and dependencies as are mentioned in the design. When an implementation closely resembles a design we say there is a high correspondence. Thus, maintaining correspondence is the effort to keep design and implementation identical.

Question 1
How would you rate the importance of correspondence between design and implementation?

- Not important
- Somewhat important
- Important
- Very important
- Extremely important

Question 2
What methods are used in your project to maintain the correspondence between UML designs and the implementation? Please choose all that apply.

- No special effort spent on maintaining correspondence
- Manually review and update the model
- Reverse engineer the implementation code
- Using round-trip engineering technique

Question 3
How frequently do you maintain the correspondence between UML designs and the implementation?

- No special effort spent on maintaining correspondence
- After final software release
- On monthly basis
- On weekly basis
- Continuously during coding activities

Question 4
How important is the correspondence between an implementation and the associated UML design in your projects?

- Unimportant
- Fairly important
- Important
- Very important
- Extremely important

Question 5
In implementing a UML design, please mark how strict you think the following statements should apply (Loosely - Somewhat Loosely - Neutral - Somewhat Strict - Strictly).

- The package structure in an implementation should correspond to the package structure in the design.
- The dependencies between classes in an implementation should correspond to the dependencies between classes in the design.
- The inheritance relations in an implementation should correspond to the inheritance relations in the design.
- The names of classes and methods in an implementation should correspond to names in the design.
- The order of method calls in an implementation should correspond to the order of messages in the design.

The purpose of a software design is to specify a system that is going to be developed. We regard a design as having a high degree of completeness if it specifies all parts of a system.

Question 6
Based on your experience, how would you (on average) rate the degree of completeness of the provided UML designs in describing the systems to be developed?

- Very Low

- Low
- Somewhat Low
- Somewhat High
- High

Question 7
In your experience, what is the frequency in which programmers go back to the designers to ask for clarification of the design in projects that use UML compared to projects that do not use UML?

- Very less often
- A bit less often
- Neutral
- Somewhat more often
- Much more often

Question 8
In case there is an inconsistency in a UML design (e.g. a class that is never called in any sequence diagram) that you have to implement, what would you do as a programmer?

- Implement the class as specified
- Discuss the inconsistency with other colleagues
- Infer the role of that class and make it functional to other classes
- Clarify the inconsistency to the designer
- Leave out the class

Question 9
Please indicate how frequent the following factors force you to deviate from a UML design in an implementation.

	Never	Sometimes	Often	Very Often
Meeting approaching deadline				
Impractical design				
Incomplete design				
Design does not satisfy requirements				

In modeling a system, designers can decide to specify different parts in various levels of detail. We say a part of a system as being modeled in high level of detail if all aspects (e.g. structures, behaviors, interactions) of that part are explicitly specified in the design.

Question 10
As a programmer, indicate how you would prefer that detail is used in UML models of a system that you have to implement (Disagree - Somewhat Disagree - Neutral - Somewhat Agree - Agree).

- All parts of a system should be specified in an equal amount of detail.
- Different parts of a system should be specified in a level of detail that is proportional to the complexity of the parts being modeled.
- Parts that are more critical for the functioning of the system should be specified in more detail.
- A model should explain how the system works, but allow programmers freedom to determine implementation details.

Question 11
Does the use of UML help or hinder you in being more productive in different activities of software development? Please indicate by marking a circle in each row of the following table.

(1 = Very Hinders; 2 = Hinders; 3 = Somewhat hinders; 4 = Neutral; 5 = Somewhat helpful; 6 = Helpful; 7 = Very Helpful)

	1	2	3	4	5	6	7
Analysis/Problem Understanding							
Design							
Implementation							
Testing							
Maintenance							

Imperfections in UML designs may appear in various forms. Inconsistency reveals contradicting information in portraying a design concern. Understandability relates to the degree in which concepts are easily inferred from a design. Inaccuracy relates to the lack of preciseness in specifying design concerns. Finally, incompleteness refers to a design's low coverage in specifying all parts of a system.

Question 12
How often do the imperfections in UML models listed below lead to problems in implementing a design?

	Never	Sometimes	Often	Very Often
Inconsistency				
Understandability				
Inaccuracy				
Incompleteness				

Question 13

Please indicate how the use of UML influences the following quality properties of the final software product.

	Reduce	Somewhat Reduce	Neutral	Somewhat Improve	Improve
Satisfying / covering Requirements					
Correctness (number of defects)					
Modularity					
Testability					
Understandability					

Your professional background is very useful for us in interpreting your answers accurately. Therefore, please continue with the following questions that concern your professional background.

Question 14
What is the highest education degree you received?

- None
- B.Sc / HBO (Dutch)
- M.Sc. / Drs. (Dutch)
- Ph.D. / Dr. (Dutch)

Question 15
Please indicate the number of year you have been working as a programmer.

- Less than 2 years
- 2 – 5 years
- More than 5 years

Question 16

Please indicate the number of software project in which you were involved as a programmer over the past 10 years.

- Less than 4 projects
- 4 – 6
- 7 - 10 projects
- More than 10 projects

Question 17
Choose the programming languages in which you have more than 2 years of experience.

- C
- C#
- C++
- Java
- Basic
- Cobol
- Other, ----------

Question 18
Indicate the location where you are currently working as a programmer.

- Netherlands
- India
- United Kingdom
- Unites States
- Other, ----------

Question 19
How have you learnt UML?

- I am not familiar with UML
- From course(s) in industry
- From course(s) at university
- Self-study from books

Question 20
What percentage of software projects that you were involved in over the past 10 years used UML?

- Less then 25 % of the projects
- 25 % of the projects
- 50 % of the projects
- 75 % of the projects
- More than 75 % of the projects

Appendix B

Model Comprehension Questionnaire

Q1) According to the diagram, there are two categories of item, i.e., ReferenceItem and LoanItem. Which of the following statements is true about those two items?
a. ReferenceItem is not allowed to be borrowed
b. LoanItem is not allowed to be borrowed
c. Both ReferenceItem and LoanItem are allowed to be borrowed
d. Both ReferenceItem and LoanItem are not allowed to be borrowed
e. Cannot determine based on the model

Q2) In the process of lending a book, the system must first check the type of the requested book item. What information is required as an input to perform this validation?
a. isbn
b. itemId
c. userId
d. title
e. Cannot determine based on the model

Q3) According to the diagram, a loan of a book cannot be renewed (extended) when other borrower has reserved it. Which of the following classes hold the information about the number of reservations of a book title?
a. Title
b. Reservation
c. Loan
d. LoanItem
e. Cannot determine based on the model

Q4) When adding an item to the system, object item is created. Besides ItemControl, the creation of object item implies its association with which of the following class objects? a. MonitorControl

b. Title
c. TitleControl
d. Loan
e. Cannot determine based on the model

Q5) One of the conditions that has to be satisfied before removing a user from the system is that the user must not have any outstanding fine/charge. From which of the following classes can object UserControl obtains this information?
a. Title
b. LoanControl
c. Loan
d. User
e. Cannot determine based on the model

Q6) What is the main functionality of class UserTerminal?
a. It provides user interface functionality to class User
b. It provides user interface functionality to class Title
c. It provides user interface functionality to class Loan
d. It provides user interface functionality that is common to all system users
e. Cannot determine based on the model

Q7) Which of the following describes the functionality of the controller classes (e.g., TitleControl, UserControl, etc.)?
a. Providing logic for all system functionality
b. Providing logic for database functionality
c. Providing logic for interacting with system users
d. Configuring or managing for the user interface functionality
e. Cannot determine based on the model

Q8) When an item of a book title is borrowed or returned, the system has to immediately update the status of this book item. Which of the following classes holds this information (i.e., status)?
a. Title
b. Loan
c. LoanItem
d. Item
e. Cannot determine based on the model

Q9) When a borrower returns a book, the system has to check whether it is returned before the due date. Which of the following classes holds the due date information of a borrowed book?
a. Title
b. User
c. Loan
d. LoanItem
e. Cannot determine based on the model

Q10) A borrower will be charged when returning books later than their due dates. Which of the following classes calculates the amount of charge for a particular loan?
a. Title
b. User
c. Loan
d. LoanItem
e. Cannot determine based on the model

Q11) In order to make a reservation of a book title in the system, which of the following classes must instantiate a Reservation object?
a. Title
b. ReservationControl
c. BorrowerTerminal
d. User
e. Cannot determine based on the model

Q12) According to the diagram, which of the following statement is TRUE with regard to scenario Remove Item?
a. The status of item must be first assessed in class Item
b. The status of item must be first assessed in class LoanItem
c. An item can be deleted regardless of its status
d. An item can be deleted only if the corresponding title is deleted
e. Cannot determine based on the model

Q13) According to the diagram, which of the following statement is TRUE with regard to scenario Remove Item?
a. The status of item must be first assessed in class Item
b. The status of item must be first assessed in class LoanItem
c. An item can be deleted regardless of its status
d. An item can be deleted only if the corresponding title is deleted
e. Cannot determine based on the model

Q14) Class LibrarianTerminal implements method removeTitle. According to the diagram, which of the following pseudo code represents the implementation of method removeTitle?
a.

```
Class LibrarianTerminal {
    TitleControl a;
    ...
    Function removeTitle(isbn) {
        a.removeTitle(isbn);
    }
    ...
}
```

b.

```
Class LibrarianTerminal {
```

```
    TitleControl a;
    ...
    Function removeTitle(isbn) {
        a.removeItem(isbn);
        a.removeTitle(isbn);
    }
    ...
}
```

c.

```
Class LibrarianTerminal {
    ItemControl a;
    TitleControl b;
    ...
    Function removeTitle(isbn) {
        create a;
        a.removeItem(itemId);
        b.removeTitle(isbn);
    }
    ...
}
```

d.

```
Class LibrarianTerminal {
    TitleControl a;
    ItemControl b;
    ...
    Function removeTitle(isbn) {
        create b;
        a.removeTitle(isbn);
        b.removeItem(isbn);
    }
    ...
}
```

e. Cannot determine based on the model

Q15) According to the diagram, which of the following indicates the correct implementation of making a book reservation (instantiating a reservation object)?

a.

```
Class ReservationControl {
    Reservation a;
    User b;
    ...
    Function requestMakeReservation(isbn) {
        create a;
```

```
        b.reservationAdded();
    }
    ...
}
```

b.

```
Class Title {
    Reservation a;
    ...
    Function requestMakeReservation(isbn) {
        create a;
        ...
    }
    ...
}
```

c.

```
Class ReservationControl {
    Reservation a;
    ...
    Function requestMakeReservation(isbn) {
        create a;
    }
    ...
}
```

d.

```
Class BorrowerTerminal {
    Reservation a;
    ...
    Function requestMakeReservation(isbn) {
        create a;
    }
    ...
}
```

e. Cannot determine based on the model

Appendix C

Descriptive Statistics of the NS-OFI Data Set

Table C.1 and C.2 provides descriptive statistics of Java classes that are modeled in class diagrams and sequence diagram respectively, when no sampling method is performed: defect types are not taken into account.

Table C.1: Descriptive statistics of classes modeled in class diagrams – NS-OFI data set

Measures	N	Med	Mean	SDev	Min	Max
CD_{aop}	48	0.00	0.32	0.44	0.00	1.00
CD_{asc}	48	0.16	0.44	0.47	0.00	1.00
Coupling	48	15.00	18.89	18.22	2.00	119.00
Complexity	48	14.00	38.20	56.34	0.00	297.00
Size (KSLoC)	48	0.24	0.49	0.80	0.02	5.26
Defect Density	48	12.47	19.06	14.88	2.67	58.82

Table C.2: Descriptive statistics of classes modeled in sequence diagrams – NS-OFI data set

Measures	N	Med	Mean	SDev	Min	Max
SD_{obj}	61	2.00	1.97	0.05	1.66	2.00
SD_{msg}	61	1.88	1.72	0.57	0.30	2.56
Coupling	61	13.00	17.96	17.51	5.00	119.00
Complexity	61	24.00	46.24	66.57	0.00	366.00
Size (KSLoC)	61	0.26	0.56	1.10	0.02	7.01
Defect Density	61	12.35	26.09	39.99	1.73	217.39

Bibliography

[1] SPSS statistical analysis tool. http://www.spss.com.

[2] SPSS Statistics Release 17.0.0 for Macintosh.

[3] The MetricView Project. http://www.win.tue.nl/empanada/metricview/.

[4] The UML design quality metrics tool. http://www.sdmetrics.com.

[5] *OMG Unified Modeling Language (OMG UML) – Superstructure version 2.2*, 2009.

[6] Abdurazik, A., and Offutt, J. Using UML collaboration diagrams for static checking and test generation. In *UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings* (2000), A. Evans, S. Kent, and B. Selic, Eds., vol. 1939, Springer, pp. 383–395.

[7] Abreu, F., and Carapuça, R. Object-oriented software engineering: Measuring and controlling the development process. In *proceedings of the 4th International Conference on Software Quality* (1994), Citeseer.

[8] Alemán, J., and Álvarez, A. Can intuition become rigorous? Foundations for UML model verification tools. In *Proceedings of the 11th International Symposium on Software Reliability Engineering (ISSRE 2000)* (2000), pp. 344–355.

[9] Ambler, S. *The elements of UML 2.0 style*. Cambridge University Press, 2005.

[10] Arisholm, E., and Briand, L. C. Predicting fault-prone components in a java legacy system. In *ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering* (New York, NY, USA, 2006), ACM, pp. 8–17.

[11] Arisholm, E., Briand, L. C., Hove, S. E., and Labiche, Y. The impact of UML documentation on software maintenance: An experimental evaluation. *IEEE Transactions on Software Engineering 32*, 6 (2006), 365–381.

[12] Arisholm, E., Briand, L. C., and Johannessen, E. B. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. Tech. rep., Simula Research Laboratory, 2008.

[13] Basanieri, F., Bertolino, A., and Marchetti, E. The cow_suite approach to planning and deriving test suites in uml projects. *UML 2002: The Unified Modeling Language* (2002), 275–303.

[14] Basili, V. The role of experimentation in software engineering: past, current, and future. In *Proceedings of the 18th international conference on Software engineering* (1996), IEEE Computer Society Washington, DC, USA, pp. 442–449.

[15] Basili, V. R., Briand, L. C., and Melo, W. L. A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Softw. Eng. 22*, 10 (1996), 751–761.

[16] Bernard, H. R., Killworth, P., Kronenfeld, D., and Sailer, L. The problem of informant accuracy: The validity of retrospective data. *Annual Review of Anthropology 13* (1984), 495–517.

[17] Boehm, B., and Basili, V. Software defect reduction top 10 list. *Foundations of empirical software engineering: the legacy of Victor R. Basili* (2005), 426.

[18] Boehm, B. W., Brown, J. R., and Lipow, M. Quantitative evaluation of software quality. In *ICSE '76: Proceedings of the 2nd international conference on Software engineering* (Los Alamitos, CA, USA, 1976), IEEE Computer Society Press, pp. 592–605.

[19] Booch, G. *Object solutions: managing the object-oriented project.* Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA, 1995.

[20] Bowerman, B. L., and O'Connell, R. *Linear Statistical Models: An Applied Approach*, 2nd ed. Thomson Learning, 1990.

[21] Briand, L., Daly, J., Porter, V., and Wust, J. A comprehensive empirical validation of design measures for object-oriented systems. In *Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International* (1998), pp. 246–257.

[22] Briand, L., Devanbu, P., and Melo, W. An investigation into coupling measures for C++. In *ICSE '97: Proceedings of the 19th international conference on Software engineering* (New York, NY, USA, 1997), ACM Press, pp. 412–421.

[23] Briand, L., and Labiche, Y. A UML-based approach to system testing. *Software and Systems Modeling 1*, 1 (2002), 10–42.

[24] Briand, L., Melo, W., and Wüst, J. Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects. *IEEE Transactions on Software Engineering* (2002), 706–720.

[25] Briand, L. C., Bunse, C., Daly, J. W., and Differding, C. An experimental comparison of the maintainability of object-oriented and structured design documents. *Empirical Softw. Eng. 2*, 3 (1997), 291–312.

[26] Briand, L. C., and Labiche, Y. A UML-based approach to system testing. In *UML '01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools* (London, UK, 2001), Springer-Verlag, pp. 194–208.

[27] Briand, L. C., Labiche, Y., Penta, M. D., and Yan-Bondoc, H. D. An experimental investigation of formality in UML-based development. *IEEE Transactions on Software Engineering 31*, 10 (2005), 833–849.

[28] Briand, L. C., Wüst, J., Daly, J. W., and Porter, D. V. Exploring the relationship between design measures and software quality in object-oriented systems. *J. Syst. Softw. 51*, 3 (2000), 245–273.

[29] Briand, L. C., Wüst, J., Ikonomovski, S. V., and Lounis, H. Investigating quality factors in object-oriented designs: an industrial case study. In *ICSE '99: Proceedings of the 21st international conference on Software engineering* (New York, NY, USA, 1999), ACM, pp. 345–354.

[30] Brito e Abreu, F., and Melo, W. Evaluating the impact of object-oriented design on software quality. In *METRICS 1996: Proceedings of the 3rd International Software Metrics Symposium* (Los Alamitos, CA, USA, 1996), IEEE Computer Society.

[31] Cartwright, M., and Shepperd, M. An empirical investigation of an object-oriented software system. *IEEE Trans. Softw. Eng. 26*, 8 (2000), 786–796.

[32] Catal, C., and Diri, B. A systematic review of software fault prediction studies. *Expert Systems with Applications 36*, 4 (2009), 7346–7354.

[33] Cherubini, M., Venolia, G., DeLine, R., and Ko, A. J. Let's go to the whiteboard: how and why software developers use drawings. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2007), ACM, pp. 557–566.

[34] Chidamber, S. R., Darcy, D. P., and Kemerer, C. F. Managerial use of metrics for object-oriented software: An exploratory analysis. *IEEE Trans. Softw. Eng. 24*, 8 (1998), 629–639.

[35] Chidamber, S. R., and Kemerer, C. F. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng. 20*, 6 (1994), 476–493.

[36] Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., and Wong, M.-Y. Orthogonal defect classification-a concept for in-process measurements. *IEEE Transactions on Software Engineering 18*, 11 (1992), 943–956.

[37] Chillarege, R., Kao, W.-L., and Condit, R. G. Defect type and its impact on the growth curve. In *ICSE '91: Proceedings of the 13th international conference on Software engineering* (Los Alamitos, CA, USA, 1991), IEEE Computer Society Press, pp. 246–255.

[38] Clarke, E., and Wing, J. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR) 28*, 4 (1996), 626–643.

[39] Coad, P., and Yourdon, E. *Object-oriented design*. Prentice Hall, 1991.

[40] Cruz-Lemus, J., Genero, M., Morasca, S., and Piattini, M. Assessing the the understandability of UML statechart diagrams with composite states - a family of empirical studies. *Empirical Software Engineering* (2009), to appear.

[41] David, A., Moller, M., and Yi, W. Formal verification of UML statecharts with real-time extensions. *Lecture Notes in Computer Science* (2002), 218–232.

[42] Dobing, B., and Parsons, J. How UML is used. *Commun. ACM 49*, 5 (2006), 109–113.

[43] Dzidek, W., Arisholm, E., and Briand, L. A realistic empirical evaluation of the costs and benefits of uml in software maintenance. *IEEE Transactions on Software Engineering* (2008), 407–432.

[44] El Emam, K., Melo, W., and Machado, J. C. The prediction of faulty classes using object-oriented design metrics. *Journal of Systems and Software 56*, 1 (2001), 63–75.

[45] Engels, G., Heckel, R., and Kuster, J. The consistency workbench: A tool for consistency management in UML-based development. *Lecture notes in computer science* (2003), 356–359.

[46] Engels, G., K
"uster, J., Heckel, R., and Groenewegen, L. A methodology for specifying and analyzing consistency of object-oriented behavioral models. In *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering* (2001), ACM New York, NY, USA, pp. 186–195.

[47] Feldman, A., Provan, G., and Van Gemund, A. FRACTAL: Efficient fault isolation using active testing. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)* (2009), pp. 778–784.

[48] Fenton, N. E., and Neil, M. A critique of software defect prediction models. *IEEE Trans. Softw. Eng. 25*, 5 (1999), 675–689.

[49] Field, A. *Discovering Statistics Using SPSS*, 2nd ed. SAGE, London, 2005.

[50] Flaton, B. Exploring the effect of UML modeling on software quality. Master's thesis, Technische Universiteit Eindhoven, 2008.

[51] Fraikin, F., and Leonhardt, T. SeDiTeC–testing based on sequence diagrams. In *ASE '02: Proceedings of the 17th IEEE international conference on Automated software engineering* (Washington, DC, USA, 2002), IEEE Computer Society, p. 261.

[52] France, R., Evans, A., Lano, K., and Rumpe, B. The UML as a formal modeling notation. *Computer Standards and Interfaces 19*, 7 (1997), 325–334.

[53] Genero, M., Cruz-Lemus, J. A., Caivano, D., Abrahao, S., Insfran, E., and Carsí, J. A. Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A controlled experiment. In *MoDELS '08: Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems* (Berlin, Heidelberg, 2008), vol. 5301, Springer-Verlag, pp. 280–294.

[54] Genero, M., Manso, E., Visaggio, A., Canfora, G., and Piattini, M. Building measure-based prediction models for UML class diagram maintainability. *Empirical Software Engineering 12*, 5 (10 2007), 517–549.

[55] Genero, M., Piattini, M., and Calero, C. A survey of metrics for UML class diagrams. *Journal of Object Technology 4*, 9 (2005), 59–92.

[56] Glezer, C., Last, M., Nachmany, E., and Shoval, P. Quality and comprehension of UML interaction diagrams-an experimental comparison. *Information and Software Technology 47*, 10 (2005), 675–692.

[57] Gyimothy, T., Ferenc, R., and Siket, I. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Trans. Softw. Eng. 31*, 10 (2005), 897–910.

[58] Hailpern, B., and Tarr, P. Model-driven development: The good, the bad, and the ugly. *IBM Systems Journal 45*, 3 (2006), 451–461.

[59] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: An update. *SIGKDD Explorations 11*, 1 (2009).

[60] Hanley, J., and McNeil, B. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology 143*, 1 (1982), 29.

[61] Harrison, R., Counsell, S., and Nithi, R. Experimental assessment of the effect of inheritance on the maintainability of object-oriented systems. *J. Syst. Softw. 52*, 2-3 (2000), 173–179.

[62] Heitmeyer, C. On the need for practical formal methods, 1998.

[63] Highsmith, J., and Cockburn, A. Agile software development: The business of innovation. *Computer* (2001), 120–122.

[64] Hosmer, D. W., and Lemeshow, S. *Applied Logistic Regression*, 2nd ed. Wiley-Interscience, 2000.

[65] IEEE. IEEE standard classification for software anomalies. *IEEE Std 1044-1993* (1994).

[66] IEEE Computer Society. *IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries*. Institute of Electrical and Electronics Engineers, New York, NY, 1991.

[67] Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. Object-Oriented Software Engineering: A Use Case Driven Approach. *Harlow, Essex, England: Addison Wesley Longman* (1992).

[68] Janes, A., Scotto, M., Pedrycz, W., Russo, B., Stefanovic, M., and Succi, G. Identification of defect-prone classes in telecommunication software systems using design metrics. *Information Sciences 176*, 24 (2006), 3711–3734.

[69] Jiang, Y., Cuki, B., Menzies, T., and Bartlow, N. Comparing design and code metrics for software quality prediction. In *Proceedings of the 4th international workshop on Predictor models in software engineering* (2008), ACM New York, NY, USA, pp. 11–18.

[70] Khoshgoftaar, T., and Seliya, N. Comparative assessment of software quality classification techniques: An empirical case study. *Empirical Software Engineering 9*, 3 (2004), 229–257.

[71] Khoshgoftaar, T. M., and Allen, E. B. Ordering fault-prone software modules. *Software Quality Control 11*, 1 (2003), 19–37.

[72] Khoshgoftaar, T. M., and Seliya, N. Fault prediction modeling for software quality estimation: Comparing commonly used techniques. *Empirical Softw. Engg. 8*, 3 (2003), 255–283.

[73] Kitchenham, B. A., and Pfleeger, S. L. Principles of survey research part 2: designing a survey. *SIGSOFT Softw. Eng. Notes 27*, 1 (2002), 18–20.

[74] Koru, A. G., and Liu, H. An investigation of the effect of module size on defect prediction using static measures. *SIGSOFT Softw. Eng. Notes 30*, 4 (2005), 1–5.

[75] Krus, D. J., and Krus, P. H. Lost: Mccall's T scores: Why? *Educational and Psychological Measurement 37*, 1 (1977), 257–261.

[76] Lange, C., DuBois, B., Chaudron, M., and Demeyer, S. An experimental investigation of UML modeling conventions. In *Model Driven Engineering Languages and Systems*, vol. 4199/2006 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 27–41.

[77] Lange, C. F. J., and Chaudron, M. R. V. Managing model quality in uml-based software development. In *STEP '05: Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 7–16.

[78] Lange, C. F. J., and Chaudron, M. R. V. Effects of defects in UML models: an experimental investigation. In *ICSE '06: Proceeding of the 28th international conference on Software engineering* (New York, NY, USA, 2006), ACM Press, pp. 401–411.

[79] Lange, C. F. J., Chaudron, M. R. V., and Muskens, J. In practice: UML software architecture and design description. *IEEE Software 23*, 2 (2006), 40–46.

[80] Leung, F., and Bolloju, N. Analyzing the quality of domain models developed by novice systems analysts. In *HICSS'05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)-Track* (2005), vol. 7.

[81] Li, W., and Henry, S. Object-oriented metrics that predict maintainability. *Journal of Systems and Software 23*, 2 (1993), 111–122.

[82] Lindland, O., Sindre, G., and Solvberg, A. Understanding quality in conceptual modeling. *IEEE software 11*, 2 (1994), 42–49.

[83] Mann, H. B., and Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics 18*, 1 (1947), 50–60.

[84] McCabe, T. J. A complexity measure. *IEEE Trans. Softw. Eng. 2*, 4 (1976), 308–320.

[85] McCall, J., Richards, P., and Walters, G. Factors in software quality. *The National Technical Information Service, Springfield, VA, USA* (1977).

[86] McUmber, W., and Cheng, B. A general framework for formalizing UML with formal languages. In *International Conference on Software Engineering* (2001), vol. 23, pp. 433–442.

[87] Mellor, S., Clark, A., and Futagami, T. Model-driven development. *IEEE software 20*, 5 (2003), 14–18.

[88] Mohagheghi, P., and Dehlen, V. Developing a quality framework for model-driven engineering. In *Models in Software Engineering*, H. Giese, Ed., vol. 5002 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 275–286.

[89] Moser, R., Pedrycz, W., and Succi, G. comparative analysis of the efficiency of change metrics and static code attributes for defect. *Proceedings of the 30th international conference on* (2008), 181–190.

[90] Murphy, G. C., Notkin, D., and Sullivan, K. J. Software reflexion models: Bridging the gap between design and implementation. *IEEE Trans. Softw. Eng. 27*, 4 (2001), 364–380.

[91] Nesi, P., and Querci, T. Effort estimation and prediction of object-oriented systems. *The Journal of Systems & Software 42*, 1 (1998), 89–102.

[92] Nugroho, A. Level of detail in UML models and its impact on model comprehension: A controlled experiment. *Information and Software Technology 51*, 12 (2009), 1670–1685.

[93] Nugroho, A. Experiment materials, http://www.liacs.nl/~anugroho.

[94] Nugroho, A., and Chaudron, M. R. V. A survey of the practice of design – code correspondence amongst professional software engineers. *ESEM* (2007), 467–469.

[95] Nugroho, A., and Chaudron, M. R. V. Managing the quality of UML models in practice. In *Model-Driven Software Development: Integrating Quality Assurance*, J. Rech and C. Bunse, Eds. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2008, ch. 1, pp. 1–36.

[96] Nugroho, A., and Chaudron, M. R. V. A survey into the rigor of UML use and its perceived impact on quality and productivity. In *ESEM '08: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement* (New York, NY, USA, 2008), ACM, pp. 90–99.

[97] Nugroho, A., and Chaudron, M. R. V. Evaluating the Impact of UML Modeling on Software Quality: An Industrial Case Study. In *Model Driven Engineering Languages and Systems: 12th International Conference, Models 2009, Denver, Co, USA, October 4-9, 2009, Proceedings* (2009), Springer, p. 181.

[98] Nugroho, A., Flaton, B., and Chaudron, M. R. V. Empirical analysis of the relation between level of detail in UML models and defect density. In *Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems (MODELS)* (2008), Czarnecki, Ed., vol. 5301 of *LNCS*, Springer-Verlag, pp. 600–614.

[99] Object Management Group. http://www.omg.org/spec/UML/.

[100] Object Management Group. The Unified Modeling Language. http://www.uml.org.

[101] Offutt, J., and Abdurazik, A. Generating tests from uml specifications. *UML '99 – The Unified Modeling Language* (1999), 76–76.

[102] Ohlsson, N., and Alberg, H. Predicting fault-prone software modules in telephone switches. *IEEE Trans. Softw. Eng. 22*, 12 (1996), 886–894.

[103] Ohlsson, N., Zhao, M., and Helander, M. Application of multivariate analysis for software fault prediction. *Software Quality Control 7*, 1 (1998), 51–66.

[104] Oppenheim, A. N. *Questionnaire Design and Attitude Measurement*. Heinemann Educational Books Ltd., 1996.

[105] Ostrand, T. J., and Weyuker, E. J. How to measure success of fault prediction models. In *SOQUA '07: Fourth international workshop on Software quality assurance* (New York, NY, USA, 2007), ACM, pp. 25–30.

[106] Otero, M. C., and Dolado, J. Evaluation of the comprehension of the dynamic modeling in UML. *Information and Software Technology 46*, 1 (2004), 35–53.

[107] Perry, D., Porter, A., and Votta, L. Empirical studies of software engineering: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (2000), ACM New York, NY, USA, pp. 345–355.

[108] Pfleeger, S., and Hatton, L. Investigating the influence of formal methods. *Computer 30*, 2 (1997), 33–43.

[109] Pfleeger, S., and Kitchenham, B. Principles of survey research: part 1: turning lemons into lemonade. *ACM SIGSOFT Software Engineering Notes 26*, 6 (2001), 16–18.

[110] Purchase, H. C., Colpoys, L., McGill, M., Carrington, D., and Britton, C. UML class diagram syntax: an empirical study of comprehension. In *APVis '01: Proceedings of the 2001 Asia-Pacific symposium on Information visualisation* (Darlinghurst, Australia, Australia, 2001), Australian Computer Society, Inc., pp. 113–120.

[111] Ricca, F., Penta, M. D., Torchiano, M., Tonella, P., and Ceccato, M. The role of experience and ability in comprehension tasks supported by UML stereotypes. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 375–384.

[112] Rosenberg, J. Some misconceptions about lines of code. In *METRICS '97: Proceedings of the 4th International Symposium on Software Metrics* (Washington, DC, USA, 1997), IEEE Computer Society, p. 137.

[113] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. *Object oriented modeling and design*. Prentice Hall, Book Distibution Center, 110 Brookhill Drive, West NYACK, NY 10995-9901 (USA), 1991.

[114] Runeson, P., and Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering 14*, 2 (2009), 131–164.

[115] Rutherford, A. *Introducing ANOVA and ANCOVA: a GLM approach*. SAGE, 2001.

[116] Selic, B. The pragmatics of model-driven development. *IEEE software 20*, 5 (2003), 19–25.

[117] Selic, B. Model-driven development: Its essence and opportunities. In *ISORC '06: Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06)* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 313–319.

[118] Shatnawi, R., and Li, W. The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process. *The Journal of Systems & Software 81*, 11 (2008), 1868–1882.

[119] Sjøberg, D., Hannay, J., Hansen, O., Kampenes, V., Karahasanovic, A., Liborg, N., and Rekdal, A. A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering 31*, 9 (2005), 733–753.

[120] Staron, M., Kuzniarz, L., and Wohlin, C. Empirical assessment of using stereotypes to improve comprehension of UML models: a set of experiments. *J. Syst. Softw. 79*, 5 (2006), 727–742.

[121] Subramanyam, R., and Krishnan, M. S. Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. *IEEE Trans. Softw. Eng. 29*, 4 (2003), 297–310.

[122] Succi, G., Pedrycz, W., Stefanovic, M., and Miller, J. Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics. *Journal of Systems and Software 65*, 1 (2003), 1–12.

[123] Tichy, W. Should computer scientists experiment more? *COMPUTER 31*, 5 (1998), 32–40.

[124] Torchiano, M. Empirical assessment of UML static object diagrams. *International Workshop on Program Comprehension* (2004), 226–230.

[125] Traore, I., and Aredo, D. Enhancing structured review with model-based verification. *IEEE Transactions on Software Engineering* (2004), 736–753.

[126] van den Broek, R. Visualizing the level of detail in UML models, 2010. Bachelor's Thesis - LIACS, Leiden University.

[127] van Opzeeland, D., Lange, C. F. J., and Chaudron, M. R. V. Quantitative techniques for the assessment of correspondence between uml designs and implementations. In *9th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2005)* (2005).

[128] Wang, X. An empirical study on the relation between the quality of UML models and the quality of the implementation. Master's thesis, Leiden Institute of Advanced Computer Science - Leiden University, December 2008.

[129] Wing, J. A specifier's introduction to formal methods. *Computer 23*, 9 (1990), 8–10.

[130] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[131] Wong, K., and Sun, D. On evaluating the layout of UML class diagrams for program comprehension. In *IWPC '05: Proceedings of the 13th International Workshop on Program Comprehension* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 317–326.

[132] Zelkowitz, M., and Wallace, D. Experimental Models for Validating Technology. *Computer 31*, 5 (1998), 23–31.

[133] Zhao, M., Wohlin, C., Ohlsson, N., and Xie, M. A comparison between software design and code metrics for the prediction of software fault content. *Information and Software Technology 40*, 14 (1998), 801–809.

[134] Zhou, Y., and Leung, H. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Transactions on Software Engineering 32*, 10 (2006), 771–789.