Model-Driven Engineering of Self-Adaptive Software

UCT CS Colloquium

University of Cape Town, South Africa, 19th August 2015

Thomas Vogel

@tomvog

System Analysis and Modeling Group
Hasso Plattner Institute
University of Potsdam, Germany

HPI

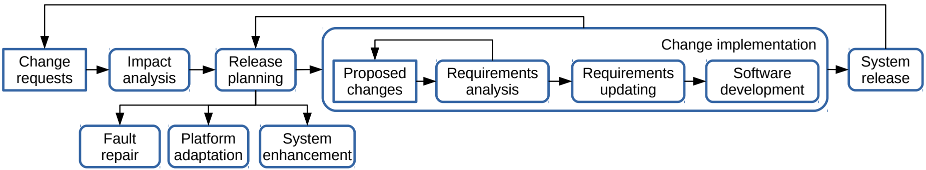
Continuous Change

- **Software aging** [Parnas, 1994]
  - When not being adapted to changing user needs (lack of movement)
  - Adapting the software often violates the design (ignorant surgery)
- **Lehman's laws of software evolution** (real-world applications)
  [Lehman and Belady, 1985, Lehman and Ramil, 2001]
  - I. A "system must be continually adapted else it becomes progressively less satisfactory in use"
  - VI. "The functional capability of [...] systems must be continually increased to maintain user satisfaction over the system lifetime"

⇒ **Software Evolution and Maintenance**

[Mens and Demeyer, 2008, Mens et al., 2010, Mens et al., 2014]

Software Evolution Process [Sommerville, 2010]

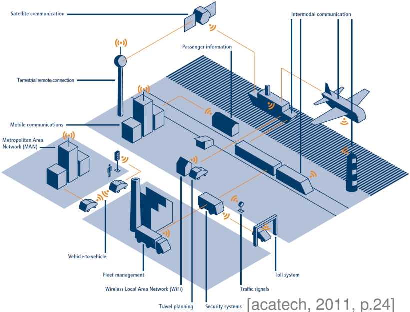


- Performed by different groups of people (support staff, developers,...) [Kitchenham et al., 1999]
- Follows a higher-level management process [Kitchenham et al., 1999]
- Enacting a release during scheduled system downtimes (stop-and-go maintenance) [Pezzè, 2012]

⇒ **Process is costly, introduces delays, and affects availability**

Software systems that are...

- context-aware (pervasive computing [Weiser, 1991, Satyanarayanan, 2001], internet of things [Perera et al., 2014])
  - timely changes
  - individual changes
- mission-critical/dependable [Shaw, 2002]
  - high or permanent availability
- complex (ultra-large-scale [Northrop et al., 2006] system of systems [Valerdi et al., 2008])
  - costs
  - dynamic integration
  - shutdown not feasible
- ...



[acatech, 2011, p.24]
distributed traffic management

⇒ Efforts and feasibility of traditional software evolution process?

⇒ **Built-in evolution/adaptation process?**

Self-Adaptive Software [Cheng et al., 2009, de Lemos et al., 2013]

"systems that are able to modify their behavior and/or structure in response to their perception of the environment and the system itself, and their goals" *[de Lemos et al., 2013, p. 1]*

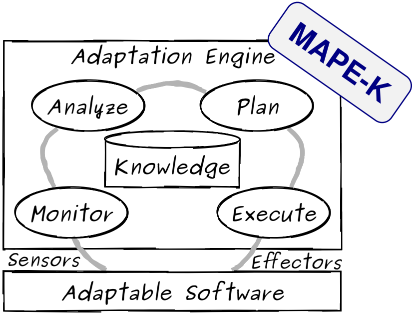
Observations:

- Self-*: configuring / optimizing / healing / protecting / managing / ...
- Shift responsibility for adaptation from developers to the system
- Shift software engineering activities from dev. time to runtime
- Blurring boundary between development time and runtime

Goal:

- Automated and dynamic adaptation
- Mitigating the growing costs, complexity, and diversity of adaptation

Feedback Loop [Kephart and Chess, 2003, Brun et al., 2009]



- Often inspired by control theory [Filieri et al., 2015]
- Turns an open-loop into a closed-loop system [Salehie and Tahvildari, 2009]
- **Architectural** blueprint: separating domain and adaptation concerns
  - Similar to computational reflection [Maes, 1987]
- Knowledge: policies and a representation (reflection) of the adaptable software [Huebscher and McCann, 2008]
  - e.g., event-condition-action rules and an architectural representation

Engineering Self-Adaptive Software

State of the Art

- Aims for reducing development efforts
- Typically, **frameworks** for feedback loops
  - Customization such as injecting policies and a representation
  - Partial generation of feedback loops based on policies

Some Drawbacks

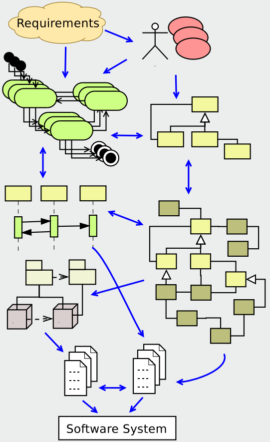
- No explicit specification and design of the feedback loops
- Closed approaches
  - Prescribe the structure and number of feedback loops
  - Restrict the techniques/types of knowledge (policies, representation,...)
- Gap between the development and runtime environments

Engineering Self-Adaptive Software with EUREMA

Side note: Model-Driven Engineering (MDE)

"The term Model-Driven Engineering (MDE) is typically used to describe software development approaches in which abstract models of software systems are created and systematically transformed to concrete implementations."

[France and Rumpe, 2007]



Engineering Self-Adaptive Software with EUREMA

Side note: Model-Driven Engineering (MDE)

Goals [France and Rumpe, 2007]

- Mitigating the gap between the problem and solution space
  - Avoiding accidental complexity of closing the gap manually
- Raise the level of abstraction (domain-specific languages & models)
- Automating development: transformation and generation
- Early analysis and quality assurance

Promises

- “Industrializing” software development [Greenfield and Short, 2003]
- Improve developers’ productivity and software quality
- Reduce costs and time to market

Engineering Self-Adaptive Software with EUREMA

Side note: Model-Driven Engineering (MDE)

"In our broad vision of MDE, models [...] are also the primary means by which developers and other systems understand, interact with, configure and modify the runtime behavior of software."

[France and Rumpe, 2007]

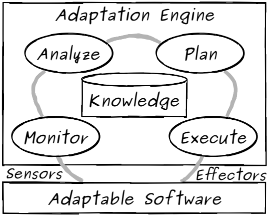
Goals of "runtime models"

- Abstractions of runtime phenomena
- Automate runtime adaptation
- Analyze running software systems

EUREMA (Executable Runtime Megamodels)

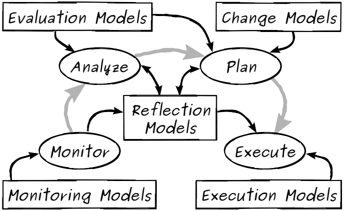
Domain-specific modeling language

- Uses feedback loop concepts
  - MAPE activities, runtime models, ...
- Explicit design of feedback loops
- Allows freely modeling feedback loops
  - Structure and number of loops
  - Techniques and types of models



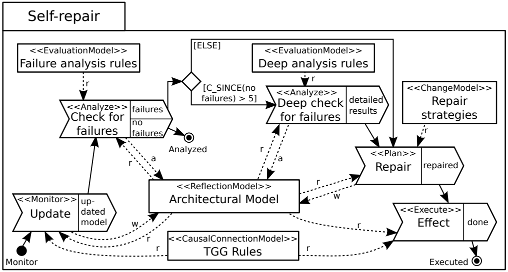
Runtime Interpreter

- EUREMA models are kept alive at runtime
- Directly executed by the interpreter
- No generation/translation steps
  - No gap between dev. and runtime env.
- Flexibility to adapt feedback loops



Language Overview

- Graphical modeling language
- Two kinds of diagrams

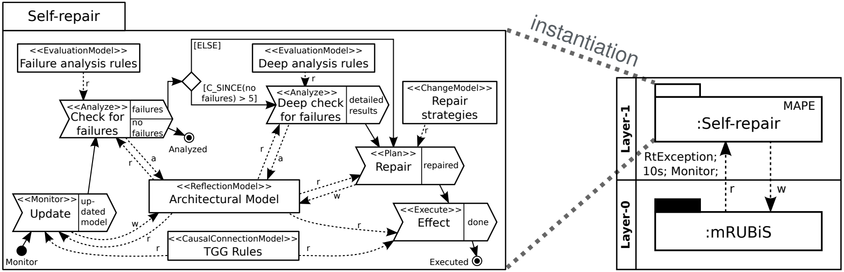


Feedback Loop Diagram (FLD)

- FLD: activities + control flow, runtime models + their usage (behavior)

Language Overview

- Graphical modeling language
- Two kinds of diagrams

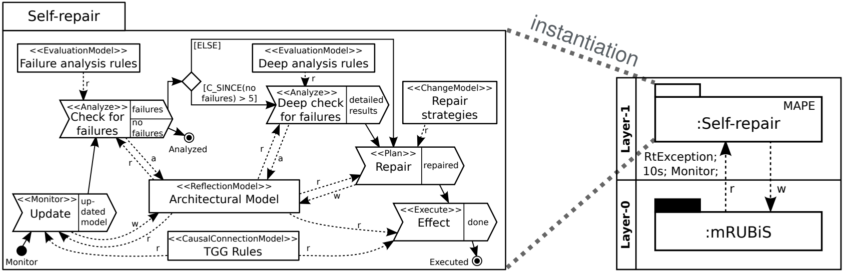


Feedback Loop Diagram (FLD) **Layer Diagram (LD)**

- FLD: activities + control flow, runtime models + their usage (behavior)
- LD: layers, white/black-box modules + their relationships (structure)
  - Trigger of modules: `<events>;<period>;<initialState>;`

Language Overview

- Graphical modeling language
- Two kinds of diagrams

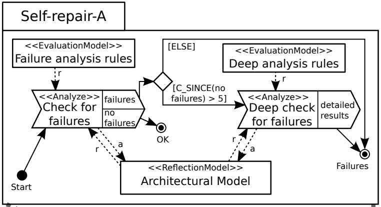


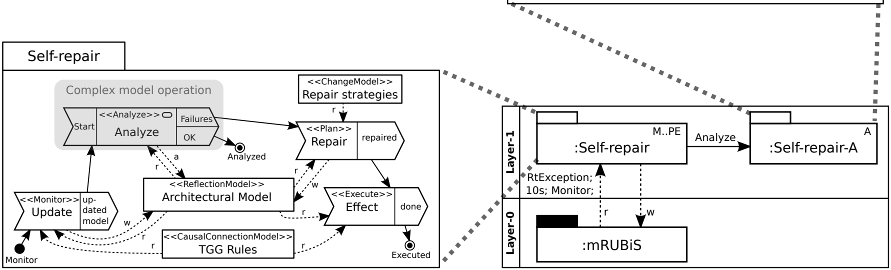
Feedback Loop Diagram (FLD) **Layer Diagram (LD)**

- FLD: activities + control flow, runtime models + their usage (behavior)
- LD: layers, white/black-box modules + their relationships (structure)
  - Trigger of modules: `<events>;<period>;<initialState>;`
- **FLDs and LD are kept alive at runtime and executed by an interpreter**

Modularity

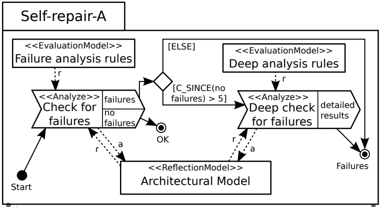
- Multiple FLDs for one feedback loop
- **Complex model operation** to invoke an FLD (entries and exists)
- Binding in the LD

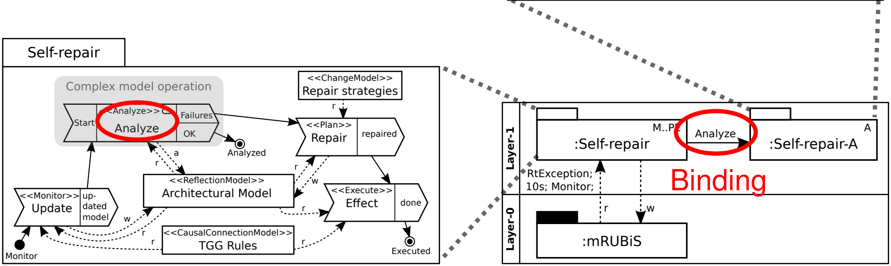




Modularity

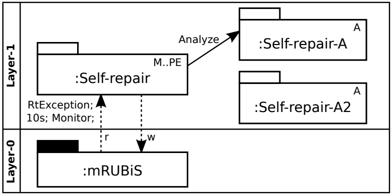
- Multiple FLDs for one feedback loop
- **Complex model operation** to invoke an FLD (entries and exists)
- Binding in the LD



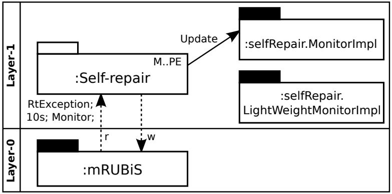


Variability

- Alternative modules as **variants**
- Rebinding to switch between alternatives
- Design-time and runtime
- Example: different analysis techniques

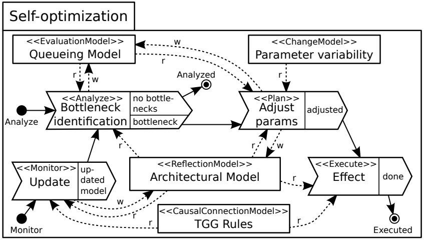


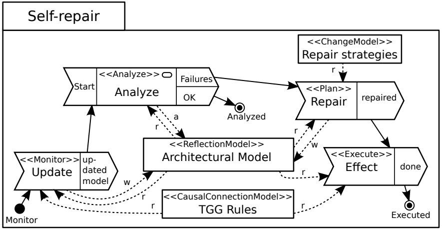
- The same applies to implementations (black-box modules) of basic model operations
- Example: different monitoring techniques



Multiple Feedback Loops

- Multiple concerns to be managed
- Competing concerns and interferences ⇒ **coordination**



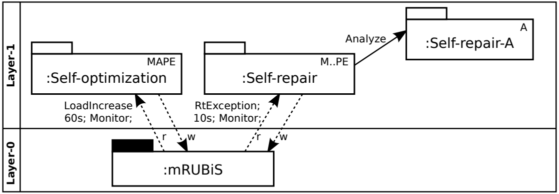


EUREMA

- Modeling the synchronized execution of feedback loops
- Model operation implementation realizes the coordination mechanism (e.g., utility functions or voting)

Multiple Feedback Loops II

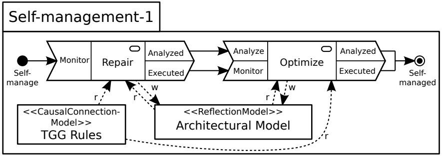
Independent execution

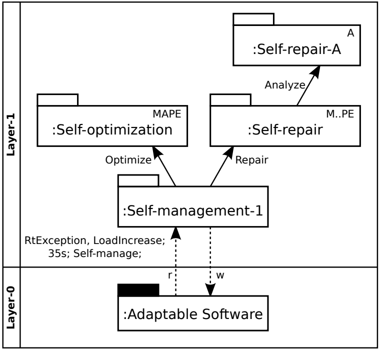


- Individual trigger for each feedback loop
- Potentially, concurrent execution of different feedback loops
- Possibility to implicitly synchronize the execution by triggers (e.g., appropriate frequencies of execution runs)

Multiple Feedback Loops III

Sequencing Complete Feedback Loops

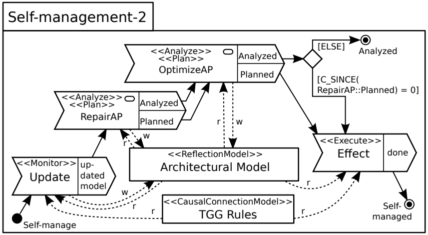


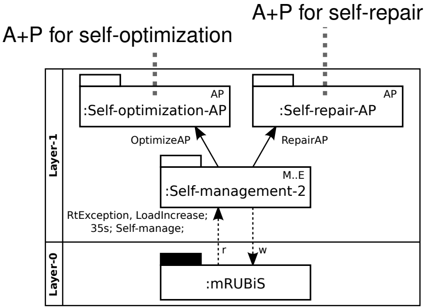


- Explicitly modeling the synchronized execution
- MAPE for self-repair → MAPE for self-optimization

Multiple Feedback Loops IV

Sequencing Adaptation Activities of Feedback Loops



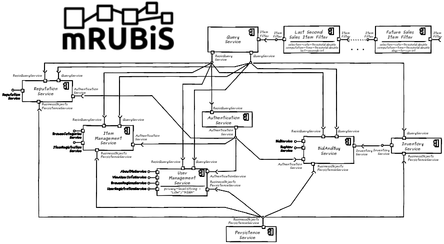


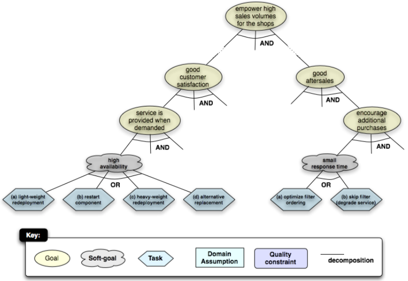
- More fine-grained synchronization (activities vs. whole feedback loop)
- Interleaved execution of different feedback loops
- M $\rightarrow$ A+P for self-repair $\rightarrow$ A+P for self-optimization $\rightarrow$ E

Evaluation

- mRUBiS as a playground
- Two cases
  - Self-healing
  - Self-optimization
- Compare alternative solutions
  - Models vs. code
  - State- vs. event-based loops

  with respect to
  - Development costs
  - Runtime efficiency
- Applied EUREMA to other approaches
  - Rainbow, DiVA, PLASMA





Conclusion

Summary and contributions of EUREMA

1. Integrated MDE approach
2. Open approach
3. Seamless Integration of Development and Runtime Environment
4. Adaptation and Evolution of Feedback Loops
5. State- and Event-Based Feedback Loops

Future Work

- Distributed feedback loops and decentralized adaptation
- Concurrent execution of interdependent feedback loops
- Model-based techniques to analyze and test EUREMA models

References I

[acatech, 2011] acatech (2011).
Cyber-physical systems: Driving force for innovation in mobility, health, energy and production.
acatech (National Academy of Science and Engineering) Position Paper,
http://www.acatech.de/de/publikationen/stellungnahmen/acatech/detail/artikel/cyber-physical-systems.html.

[Brun et al., 2009] Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., and Shaw, M. (2009).
Engineering Self-Adaptive Systems through Feedback Loops.
In Cheng, B. H., de Lemos, R., Giese, H., Inverardi, P., and Magee, J., editors, *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science (LNCS)*, pages 48–70. Springer.

[Cheng et al., 2009] Cheng, B. H., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Serugendo, G. D. M., Dustdar, S., Finkelstein, A., Gacek, C., Geihs, K., Grassi, V., Karsai, G., Kienle, H. M., Kramer, J., Litoiu, M., Malek, S., Mirandola, R., Müller, H., Park, S., Shaw, M., Tichy, M., Tivoli, M., Weyns, D., and Whittle, J. (2009).
Software Engineering for Self-Adaptive Systems: A Research Roadmap.
In Cheng, B. H., de Lemos, R., Giese, H., Inverardi, P., and Magee, J., editors, *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science (LNCS)*, pages 1–26. Springer.

[de Lemos et al., 2013] de Lemos, R., Giese, H., Müller, H., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N. M., Vogel, T., Weyns, D., Baresi, L., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Desmarais, R., Dustdar, S., Engels, G., Geihs, K., Goeschka, K., Gorla, A., Grassi, V., Inverardi, P., Karsai, G., Kramer, J., Lopes, A., Magee, J., Malek, S., Mankovskii, S., Mirandola, R., Mylopoulos, J., Nierstrasz, O., Pezzè, M., Prehofer, C., Schäfer, W., Schlichting, R., Smith, D. B., Sousa, J. P., Tahvildari, L., Wong, K., and Wuttke, J. (2013).
Software Engineering for Self-Adaptive Systems: A second Research Roadmap.
In de Lemos, R., Giese, H., Müller, H., and Shaw, M., editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475 of *Lecture Notes in Computer Science (LNCS)*, pages 1–32. Springer.

[Filieri et al., 2015] Filieri, A., Maggio, M., Angelopoulos, K., D?Ippolito, N., Gerostathopoulos, I., Hempel, A. B., Hoffmann, H., Jamshidi, P., Kalyvianaki, E., Klein, C., Krikava, F., Misailovic, S., Papadopoulos, A. V., Ray, S., Sharifloo, A. M., Shevtsov, S., Ujma, M., and Vogel, T. (2015).
Software Engineering meets Control Theory.
In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS'15, page tbd. IEEE.
to appear.

[France and Rumpe, 2007] France, R. and Rumpe, B. (2007).
Model-driven development of complex software: A research roadmap.
In *2007 Future of Software Engineering*, FOSE '07, pages 37–54. IEEE.

[Garlan et al., 2004] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., and Steenkiste, P. (2004).
Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure.
Computer, 37(10):46–54.

[Greenfield and Short, 2003] Greenfield, J. and Short, K. (2003).
Software factories: Assembling applications with patterns, models, frameworks and tools.
In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, pages 16–27. ACM.

[Huebscher and McCann, 2008] Huebscher, M. C. and McCann, J. A. (2008).
A survey of autonomic computing—degrees, models, and applications.
ACM Comput. Surv., 40(3):7:1–7:28.

References II

[Kephart and Chess, 2003] Kephart, J. O. and Chess, D. (2003).
The Vision of Autonomic Computing.
Computer, 36(1):41–50.

[Kitchenham et al., 1999] Kitchenham, B. A., Travassos, G. H., von Mayrhauser, A., Niessink, F., Schneidewind, N. F., Singer, J., Takada, S., Vehvilainen, R., and Yang, H. (1999).
Towards an ontology of software maintenance.
Journal of Software Maintenance: Research and Practice, 11(6):365–389.

[Lehman and Belady, 1985] Lehman, M. M. and Belady, L. A., editors (1985).
Program evolution: processes of software change.
Academic Press Professional, Inc., San Diego, CA, USA.

[Lehman and Ramil, 2001] Lehman, M. M. and Ramil, J. F. (2001).
Rules and tools for software evolution planning and management.
Ann. Softw. Eng., 11(1):15–44.

[Maes, 1987] Maes, P. (1987).
Concepts and experiments in computational reflection.
In *Conference Proceedings on Object-oriented Programming Systems, Languages and Applications*, OOPSLA '87, pages 147–155. ACM.

[Mens and Demeyer, 2008] Mens, T. and Demeyer, S., editors (2008).
Software Evolution.
Springer.

[Mens et al., 2010] Mens, T., Gueheneuc, Y.-G., Fernandez-Ramil, J., and D'Hondt, M. (2010).
Guest editors' introduction: Software evolution.
IEEE Software, 27(4):22–25.

[Mens et al., 2014] Mens, T., Serebrenik, A., and Cleve, A., editors (2014).
Evolving Software Systems.
Springer.

[Morin et al., 2009] Morin, B., Barais, O., Jézéquel, J.-M., Fleurey, F., and Solberg, A. (2009).
Models@ Run.time to Support Dynamic Adaptation.
Computer, 42(10):44–51.

[Northrop et al., 2006] Northrop, L., Feiler, P., Gabriel, R. P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K., and Wallnau, K. (2006).
Ultra-Large-Scale Systems: The Software Challenge of the Future.
Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[Parnas, 1994] Parnas, D. L. (1994).
Software aging.
In *Proceedings of the 16th International Conference on Software Engineering*, ICSE '94, pages 279–287. IEEE.

[Perera et al., 2014] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014).
Context aware computing for the internet of things: A survey.
IEEE Communications Surveys & Tutorials, 16(1):414–454.

References III

[Pezzè, 2012] Pezzè, M. (2012).
From off-Line to continuous on-line maintenance.
In *28th IEEE International Conference on Software Maintenance*, ICSM '12, pages 2–3. IEEE.

[Salehie and Tahvildari, 2009] Salehie, M. and Tahvildari, L. (2009).
Self-adaptive software: Landscape and research challenges.
ACM Trans. Auton. Adapt. Syst., 4(2):1–42.

[Satyanarayanan, 2001] Satyanarayanan, M. (2001).
Pervasive Computing: Vision and Challenges.
IEEE Personal Communications, 8(4):10–17.

[Shaw, 2002] Shaw, M. (2002).
Everyday dependability for everyday needs.
In *Supplemental Proceedings of the 13th International Symposium on Software Reliability Engineering*, ISSRE '02, pages 7–11. IEEE. (keynote).

[Sommerville, 2010] Sommerville, I. (2010).
Software Engineering.
Addison-Wesley, 9 edition.

[Tajalli et al., 2010] Tajalli, H., Garcia, J., Edwards, G., and Medvidovic, N. (2010).
Plasma: A plan-based layered architecture for software model-driven adaptation.
In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 467–476. ACM.

[Valerdi et al., 2008] Valerdi, R., Axelband, E., Baehren, T., Boehm, B., Dorenbos, D., Jackson, S., Madni, A., Nadler, G., Robitaille, P., and Settles, S. (2008).
A research agenda for systems of systems architecting.
International Journal of System of Systems Engineering, 1(1–2):171–188.

[Vogel and Giese, 2012] Vogel, T. and Giese, H. (2012).
A Language for Feedback Loops in Self-Adaptive Systems: Executable Runtime Megamodels.
In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012)*, pages 129–138. IEEE.

[Vogel and Giese, 2014] Vogel, T. and Giese, H. (2014).
Model-Driven Engineering of Self-Adaptive Software with EUREMA.
ACM Trans. Auton. Adapt. Syst., 8(4):18:1–18:33.

[Weiser, 1991] Weiser, M. (1991).
The Computer for the 21st Century.
Scientific American, 265(3):94–104.