Self-organizing Methods and Models for Software Development

Daniel J. Dubois
Advisor: Prof. Elisabetta Di Nitto
CoAdvisor: Prof. Raffaela Mirandola
Tutor: Prof. Gianpaolo Cugola

Milan – February 17th, 2011

Motivation

• Context:
  – Dynamic and distributed software architectures
  – Changing environment and requirements

• Problem:
  – Find efficient adaptation strategies to deal with the complexity of such architectures

Existing Solutions (1/2)

• Top-down adaptation approaches
  – Adding one or more adaptation layers on top of the system elements
  – Higher layers issue decisions for the lower layers

Advantages
• Efficient for small and medium systems
• Deterministic and optimal

Disadvantages
• Scalability issues due to complexity
• Difficult to deal with uncertainty

Existing Solutions (2/2)

• Bottom-up adaptation approaches
  – Adaptation emerges through simple decisions made by elements at the lower levels
  – Inspired by natural phenomena
  – Also known as Self-organization

Advantages
• Simple coordination
• Scalable
• Fault tolerant

Disadvantages
• Non-determinism
• Less guarantees
• Difficult to engineer
Research Hypotheses

Assumptions:
- Complex distributed software architectures
- High number of elements at the same hierarchical level
- High uncertainty and dynamism
- No elements alone are essential

RH1: Usefulness of Self-organization
- Self-organization is an efficient approach to address selected classes of problems under the aforementioned assumptions

RH2: Engineering Real Self-organizing Systems
- The self-organization design guidelines we propose simplify the adoption of self-organizing approaches in real systems

Approach to Demonstrate Research Hypothesis 1
- Study and analyze selected recurrent problems in distributed software architectures:
  - Load Balancing problems
  - Energy reduction problems
  - Publish/Subscribe middleware optimization problems

Problem 1: Decentralized Heterogeneous Load-balancing algorithm [SAS08, Bionetics08]

Problem 2: Energy Optimization in a Self-organizing Data Center [SOAR10]
Problem 3: Traffic Reduction in a Content-based Publish-subscribe System [ICAC09]

Publish: soccer
Subscribe: sport

Publish: soccer
Subscribe: sport
Broker 2
Broker 3
Broker 1

After Self-organization:
1 saved message

Approach to Demonstrate Research Hypothesis 2

• Study a common software engineering approach for designing self-organizing software:
  – Identify commonalities in self-organizing algorithms in terms of principles
  – Reference architecture and transposition of the principles into design patterns
  – Design guidelines and validation approaches

Self-organization Principles

• Derived from the observation of natural phenomena.
• They may be composed together and translated into real deployable algorithms.

• Most important evolutionary principles:
  – Noise → "a strategy of errors"
  – Emergence → "a team is better than the sum of its individuals"
  – Evolution → "natural selection"
• Most important communication principles:
  – Diffusion → "gossiping is not necessarily a bad habit"
  – Stigmergy → "leave a note and I’ll read it"

Reference Architecture
Identify principles/algorithms:
- Analyze the problem/requirements.
- Choose the principles/algorithms that are most suitable for that class of problem.

Building Model:
- Compose and instantiate principles/algorithms identified so far.
- Prove that the model satisfies its requirements.

Example of Model Construction and Validation: The Load Balancing case

- Abstract Model construction:
  - Graph representation
  - Transformation rules (may be non-deterministic)
  - Expressed using TRIO language (first-order temporal logic)

- Considered validation alternatives:
  - Automatic Model Checking on the TRIO model
  - Monte Carlo Simulations
Creating Self-org Algorithms for Software Systems

**Implementation:**
- Translate the model into a deployable form.
- Use the reference architecture.
- Use self-organizing design patterns based on the principles.

**Advantages**
- Code reusability.
- Separation between self-organizing aspects and application logic.

Identify and solve real-world problems not captured by the model:
- Race conditions.
- Component synchronization.
- Frequency of iterations.
- Amount of communication.
- ...

Conclusions

- In this thesis we explore the problem of applying self-organization techniques to solve problems in distributed software architectures.

- **Algorithmic contributions:**
  - Distributed Decentralized Load Balancing in Heterogeneous Networks.
  - Energy Optimization in a Self-organizing data center.

- **Methodological contributions:**
  - Examples of application of such method for implementing a self-organization algorithm in a real software architecture.
Next Steps

• Future challenges:
  – Find optimal ways to self-calibrate self-organizing algorithms to reduce the needs of human-assisted preliminary simulations.
  – Specialize the principles and the design patterns.
  – Collect design patterns into a library and make them available in existing CASE tools and in ready-to-use parameterizable implementations.
  – Extend the work with additional real case studies to support the validation of the patterns.
  – Investigate more on the validation techniques for self-organization.

Publications Related to this Work (1)

• International Conferences
  1. [Bionetics07] E. Di Nitto, D. J. Dubois, R. Mirandola “Self-aggregation algorithms for autonomic systems”. In Bionetics ’07, Budapest, Hungary.
  4. [Bionetics08] E. Di Nitto, D. J. Dubois, R. Mirandola “Applying self-aggregation to Load Balancing: Experimental results”. In Bionetics ’08, Hyogo, Japan.
  5. [ICAC09] E. Di Nitto, D. J. Dubois, R. Mirandola “Overlay self-organization for traffic reduction in MultiBroker publish-subscribe systems”. In ICAC ’09 (short paper), Barcelona, Spain.
  7. [Bionetics10] B. A. Capraraescu, N. M. Calcavecchia, E. Di Nitto, D. J. Dubois “SOS Cloud: Self-organizing Services in the Cloud”. In Bionetics ’10 (work in progress paper), Boston, USA.

Publications Related to this Work (2)

• Workshops and Doctoral Symposium

• Book chapters
ADDITIONAL SLIDES

Design Pattern for Diffusion (1/2)

Design Pattern for Stigmergy (1/2)

Design Pattern for Diffusion (2/2)

Design Pattern for Stigmergy (2/2)