Autonomic Computing Meets Complex Information Systems: Theory and Practice

Prof. A. Taleb-Bendiab
School of Computing
Liverpool John Moores University
email: a.talebbendiab@livjm.ac.uk
http://www.cms.livjm.ac.uk/taleb
Outline

- Understanding the gap between theory and practice
  - Pervasive and Trusted Network and Service Infrastructures
  - Drivers for a paradigm shift -- Software Intensive Systems
    - In the quest of better “Service and software architectures, infrastructures and engineering” models
      - Recent bio-inspired initiatives
        - SAS, DASADA, SRS and ANTS
    - A brief Introduction into Autonomic systems engineering
      - Definitions and state of the art
      - What has been done so far
  - Challenges and Open Questions
- Conclusions and Q&A
Emerging Networked Landscapes
Drivers for a Paradigm Shift #1

- Modern Expectations
  - High-Availability -- 24x7 delivery
    - near-100% availability is becoming mandatory for e-commerce, enterprise apps, online services, ISPs
  - Change
    - Support rapid deployment of new hw/sw, services, etc
  - Maintainability
    - Provide flexible systems admin. env.
      - reduce system administrators tasks, complexity and cost
  - Just-in-time scalability
    - Allow flexible system up scaling without sacrificing performance, availability or maintainability
      - evolutionary growth and adaptation
  - Survivability
  - Full malleability
Drivers for a Paradigm Shift #2

- Key question is not only how to achieve the above listed modern expectations as:
  - a single metric/attribute or a cost-effective combination of them all
    - Within multi-service provider settings and management domains
  - how to reduce the cost and complexity of achieving that

- How
  - Nature-inspired models – Autonomic Computing
    - Management by delegation
    - Etc.
Major Recent Initiatives

Some Recent Initiatives

- **SAS -- Self-Adaptive Systems** (DARPA, 1997)
- **DASADA -- Dynamic Assembly for Systems’ Adaptability, Dependability, and Assurance** (DARPA, 2000)
- **AC (IBM, 2001)**
- **Autonomic Communication (EU, 2003)**
- **SRS -- Self-Regenerative Systems** (DARPA, 2003)
- **ANTS -- Autonomous Nano-Technology Swarm** (NASA)

A 2020 vision of a class of space exploration missions termed *nanoswarms*, where many cooperating picospacecraft or intelligent spacecraft work in teams to explore the asteroid belt, based on the efficiency and coordination of hive culture.
Theoretical Background

- Has been informed by a set of design paradigms
  - Model-based vs Self-Organising Systems design models
    - Top-down vs bottom-up
  - Applying and/or revisiting:
    - cybernetic principles
      - control systems theory, regulation, reward and sanctions
    - Decision theory, Complexity theory
    - DAI and CI
      - dynamic planning, deliberative models, ML
    - Middleware support
      - self-awareness, reflection and deliberation
    - Autonomic Software Architecture, etc.
Definitions #1

- Can be defined as:

  - 3. Yet Another

  - can operate independently of (or with limited) human intervention, thus hiding their systems’ design and management complexity including intricacies of the automation of laborious administration tasks, recovery from unanticipated system’s failure, and/or self-protection from security vulnerability.
Characterising AC Capabilities

- Characterising AC Systems
  - A software system is autonomic, if it possesses the following capabilities:
    - **Self-configuring** — choosing a suitable behaviour, based on user preferences, context, ...
    - **Self-tuning** — choosing behaviours that optimize certain qualities (performance, year-end profits, ...) 
    - **Self-repairing** — shifting execution to another behaviour, given that the current one is failing
    - **Self-protecting** — choosing a behaviour that minimizes risks (attacks, viruses, ...)

---

Prof. A. Taleb-Bendiab, Keynote Lecture: WEBIST’07, Barcelona, Date: 13/09/2007, Slide: 9
**The Story So Far #1**

- State-of-the-art of autonomic systems designs including:
  - autonomic software models and architecture, standards
  - tools and techniques to support
    - the design, modelling, analysis
    - and evolution of autonomic software
- Define associated models for their
  - programming, control
  - interaction models with human and/or other non-AC systems (legacy).
- Delegation of authority,
  - its adjustment and revocation
The Story So Far #2

- Define a **base reference architecture** model which creates a common vernacular for autonomic computing
- Deliver **core infrastructure technologies** that provide for an open framework for the industry
- Deliver **products** with built-in autonomic capabilities
- Create and leverage open **standards** for autonomic computing

**Autonomic core capabilities**

- **Business policy**
- **Resource Provisioning**
- **Workload Management**
- **Solution Install**
- **Policy**
- **Admin Console**
- **Problem Determination**

**Products delivering autonomic features**

- IBM ThinkVantage™
- DB2 Information Management Software
- TotalStorage™
- e-server
- Lotus software
- Tivoli software
- WebSphere software

**Open Standards**

- OASIS
- GGF
- eclipse
- The Apache Software Foundation
Remaining Research Challenges

- Host-Based
  - Complexity Paradox
    - Autonomic computing aims to reduce admin. costs, hide system complexity and intricacy,
    - Though, their designs are becoming more complex
      - as yet are poorly understood as echoed by D. Garlan 2005
      - “… how do we design, build, and evolve such sw systems so that they can meet given—and evolving—requirements …”
        - Incremental deployment of AC capabilities in legacy systems.
        - AOP-based evolution, Interoperation
        - Support functional and non-functional requirements for autonomy.
        - Evaluation mechanisms and metrics [ref]
  - Governance vs Autonomy Paradox
    - Balancing and adjusting governance and autonomy
    - Programming, control and Interaction Models

- Usage paradox
  - Enriching and Interacting with BP
    - Process-centric interaction

- Complex-Based
  - Self-organisation
  - Complex and random Networks of autonomic system
The Story so far #3

- More recent work is focusing on scalable methods for specifying
  - dynamic behaviour of autonomic systems.
- Evolving
  - policies and control model
  - Structural/organisational model
- Bounded autonomy and adjustments
- Unifying models for
  - model-based and SOS approaches for autonomic systems engineering and management
Designing Autonomic Systems #1

- More recent theoretical work is focusing on scalable methods for specifying and enacting dynamic behaviour of autonomic systems
- Autonomic Systems Engineering
    - An autonomic manager contains a continuous control loop that monitors activities and takes actions to adjust the system to meet business objectives
    - Autonomic managers learn from past experience to build action plans
    - Elements need to be instrumented consistently, based on open standards
- Our model
  - Model-based Approach
    - Systems theory, design patterns, design grammar and service-oriented programming
    - K. Liu, A. Taleb-Bendiab, "Presenting a Case for a Principled Approach to Business and Technology Integration in e-Government Services: Challenges and Research Opportunities", Egov'05

Cybernetics: The Viable System Model

- Beer’s VSM implements a control & communication structure via hierarchies of homeostats (feedback loops) (1950)
- Defines 6 major systems ensure ‘viability’ of the system
  - Implementation S1
  - Monitoring S2
  - Audit S3*
  - Control S3
  - Intelligence S4
  - Policy S5
- Offers an extensible, recursive, model-based architecture, devolving autonomy to subsystems
Standards -- Interoperation

Figure 1. Representative standards used by an autonomic manager
AC Design Process -- correct ab initio

- Classical SE approaches
  - Top-down -- right 1st time --

- Ecosystem
  - Complexity: Cohesion
  - Seth Bullock and D. Cliff (HP Report, Ref.)
  - Complexity and Emerging Behaviour in IT Systems

Easily re-worked to resist new kinds of directed or undirected attack (Thompson, 1998; Layzell, 2001). This emphasis begins to shift the burden of design from a “right-first-time” stance to one of “continuous-redesign” (i.e., adaptation) in the face of changing circumstances. This relaxes traditional requirements of provably correct and robust designs that are required to solve fixed problems for all time, in favour of methodologies that are maximally responsive to failures that are considered an unavoidable fact of life (an approach sometimes termed “satisficing” rather than “optimising”, Simon, 1981). Similar approaches could perhaps be developed to cope with spam, worms, viruses, and system incompatibilities or conflicts, etc. Here the self-healing or self-repair abilities of natural complex adaptive systems are a direct inspiration.
Entropic Autonomic Design

Self Management Pattern Use Case Diagram

Class Diagram 1

A1: provide a computing service
A2: monitor that the computing service facilitates business activities and take control action as necessary
A3: maintain technology in use
A4: maintain stock of spare equipment, parts and supplies
A5: be aware of the cost of the computing service
A6: be aware of the benefit of the computing service
A7: maintain record of technology in use
A8: install new technology
A9: procure new technology
A10: withdraw obsolete technology
A11: be co-ordinating and incorporating
A12: be aware of technology opportunities & threats
A13: be aware of organisation needs
A14: be aware of organisation constraints
A15: develop technology support plan
A16: define expectations

Diagram Content Summary

- Coordination
- CoreOperation
- PublicOperationContainer
- PrivateOperationContainer

Diagram Content Detail

Documentation:
- @author
- Depend to:
  - CoreOperation

Autonomic computing capability

JMU
Modelling Autonomy #1

- Algebraic specification
  - Process Algebra
    - CSP
  - Static model checking and dynamic software analysis
    - Key States and Transitions are not enough
    - Intelligent organisation emerges as in natural systems

- Our model -- axiomatic
  - Using SSC used to formalizes the behaviour of dynamically changing systems FOL (McCarthy, 1963).
    - Support concurrent actions and timing constraints.
    - Each situation can be viewed as a history of previous actions.
    - Action, guards and time can be modelled at deliberation points in an autonomic setting.
Modelling Autonomy #2

- Evolving and Adjustable Autonomy
  - Via compositional model and evolution of
    - Software services (components)
  - **Software governance (Control)** via
    - Formal modelling of norms, policies
    - **Enactment support** – from spec. to code using Neptune language


**Governing Autonomy**

```plaintext
define situation RegenerationTest
{
  if (Concept["Concepts\HeavyLoad"])
  {
    return true;
  }
  else
  {
    return concept HeavyLoad as boolean
    {
      if (instrument
        ["Resource/CPULoad"].value > 60
        &&
        ["Resource/AvailableMemory"].value > 2000
      )
    }
  }
}

define action Regenerate using RegenerationTest
{
  Service s = Regenerate(me, machineID);
  Reroute(s);
}

define situation pRegenerate using RegenerationTest
{
  if (Concept["Concepts\HeavyLoad"])
  {
    return true;
  }
  else
  {
    return false;
  }
}
```
Learning Control Rules as an Emerging Behaviour
Emergence of Control Rules – Adjustable Autonomy in Action

- To run the AVI of Soot in action, click inside the window below.
  - This shows how Neptune language is used to generate and deploy a meta-system observer, which can observe the self-organising soot algorithm, and learn autonomic control rules from the system operation.
  - The control rule as action history are randomly shared with other soot agents and rewarded the more they are used – AKO distributed reinforcement learning.
  - This approach supports control rules extraction too.
  - Etc.
BP to SOA -- Dental Triage Demo.

Current System

Visual Modelling of Protocol or process flow

Current System:
- **CALLER ID**
  - Record Caller’s Name, Phone Number and Call Type

- **PATIENT ID**
  - Record Patient’s Name, DOB and Relationship to Caller

- **OWN DENTIST**
  - If yes, proceed
  - If no, record to dentist

Re-engineering via Neptune toolkit:
- New Visual Modelling of Protocol or process flow
- **Process is compiled into an open introspective format**

New Grid-Based System

New System:
- **New Grid-Based System**
2nd Scenario: Situation-Aware Decision Systems
Process-Oriented Programming

#1
Process-Oriented Programming

#3

- Process Modeller
  - Flow Model API
  - Logic Interpreter
  - Execution Manager
  - State Manager

- Semantic Linker
  - Linker
  - Task Interpreter
  - NILG Interpreter
  - Instruction Set Definition
  - Refactor Tool

- 3rd Party Modeller
- Compilation

- Neptune Framework
  - Controller
  - Process Modeller

- Neptune Connector
  -Convert XML to String and process return
  -Send Request to Controller

- BL Connector
  -Soap call to http://neptune.central method: exampleprocess with XML String

- Interface (RPC)
  - returns "<b>Title</b>" as String

- Controller
  - NeptuneConnector.ExecuteProcess("exampleProcess","string Title");

Prof. A. Taleb-Bendiab, Keynote Lecture: WEBIST'07, Barcelona, Date: 13/09/2007, Slide: 28
Petshop Benchmark

Figure 8.1: Layered Structure of PETSHOP

Figure 8.2: Original Order Process

Figure 8.3: Adapted Order Process
Neptune and *DIY* Computing #1

- **Application-centric mashup**
  - User requirements -- process

- **Our approach -- Neptune**
  - Allows for process descriptions to be mapped, computationally, to actual source code
    - Developers can write code
    - Domain experts can write ontological and domain-specific models
  - Fully open, and exportable
    - XML based
    - Suited for current social network tools
Neptune and **DIY Computing #2**

Wiki tools etc. can be used to produce user-friendly, centric services to once complex, and difficult computational tasks. Web site tools can produce models suited for Neptune. Neptune then maps these to actual code, producing a direct relationship between expert knowledge and an SOA.
Acknowledgements

- Acknowledgements
  - Sponsors and Partners
    - EPSRC
    - UK NHS (Christies and Linda McCartney NHS trusts, EL Dental Services)
    - IBM (PhD Fellowship) and WITSA (Egovernment)
  - The researchers and staff:
    - [www.cms.livjm.ac.uk/2nrich](http://www.cms.livjm.ac.uk/2nrich)
    - [www.cms.livjm.ac.uk/cloud](http://www.cms.livjm.ac.uk/cloud)
    - [www.cms.livjm.ac.uk/taleb](http://www.cms.livjm.ac.uk/taleb)
That’s the end – so I’m off!