Welcome to SENG 480B / CSC 485A / CSC 586A Self-Adaptive and Self-Managing Systems

Dr. Hausi A. Müller
Department of Computer Science
University of Victoria

http://courses.seng.uvic.ca/courses/2015/summer/seng/480a
http://courses.seng.uvic.ca/courses/2015/summer/csc/485a
http://courses.seng.uvic.ca/courses/2015/summer/csc/586a

1

Announcements

- Friday, June 19
  - A2 due
  - Groups formed today
- Grad project
  - Handed out June 15
  - Due July 25
- Midterm 1
  - Should be graded by June 18
  - Talk about answers on June 18

2

The Internet of Things (IoT)

Dr. Daniel Ford, Dell Research, Santa Clara, CA
Talk today 3:00-4:00 pm in ECS 660

Until recently, most of our civilization's computational and communications infrastructure has been built to a scale that is a rough function of the size of the human population. This includes the global telephone system, the Internet, and the size of the data centers that power Facebook and Google, among many other examples. That infrastructure was created to support the direct activities of humans as they made telephone calls, sent emails, or clicked on web pages. If the human population was much larger than it is today, all of that infrastructure would need to be larger as well; if the population was much smaller, the infrastructure all could be much smaller, as well. Today, that relationship is quickly evaporating.
This talk will introduce IoT as a departure from human centric computing in which all computations can be traced back to an action taken by a human. Instead, in the world of the IoT, humans are "out-of-the-loop," and, as such, their numbers are not a "constraint" on the growth of computational infrastructure. In the IoT, the population of devices initiating computations and requiring communications is virtually unrestricted; hundreds of Billions of network connected devices is not unrealistic. This development will have profound implications and will create many opportunities, and many issues. The talk will examine some of the technology behind the IoT, how it is likely to develop, and some of the unexpected consequences that arise. It will discuss research problems that are emerging in this area as well as some of the potential approaches to their solution.

3

Assignment 2 — Part I

Part I

In Part I you are to answer some basic questions about PID controllers. To get started read the Ziegler–Nichols Rules for Tuning PID Controllers.

http://www.engr.mun.ca/~nick/eng5951/Ziegler_Nichols.pdf

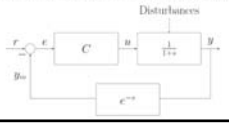
The answers for these questions should fit into Maximum 3 typeset pages.

Question 1: List any three Control Types Associated with Using the Ziegler–Nichols method.

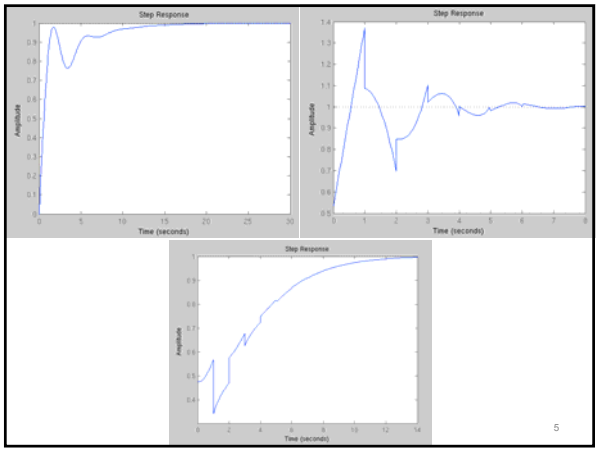
Question 2: Use the Ziegler–Nichols Method to Solve for i) classic PID ii) no overshoot PID. (where: ultimate gain Ku = 2.3, oscillation period Tu = 0.8s)

Question 3: Describe a scenario where no overshoot PID would be advantageous over classic PID.

Question 4: Use MATLAB to develop any controller for the following LTI system. Make up any controller you want and play with it until you're happy with the Step Response produced. What is your control equation C?



Questions
Ron Desmarais
rd@uvic.ca



5

Assignment 2 — Part II

Part II - Group Project (4 people max per group)

In Part II you are to design and simulate a simple PID controller. Moreover, you are to document your experience in the form of a tutorial.

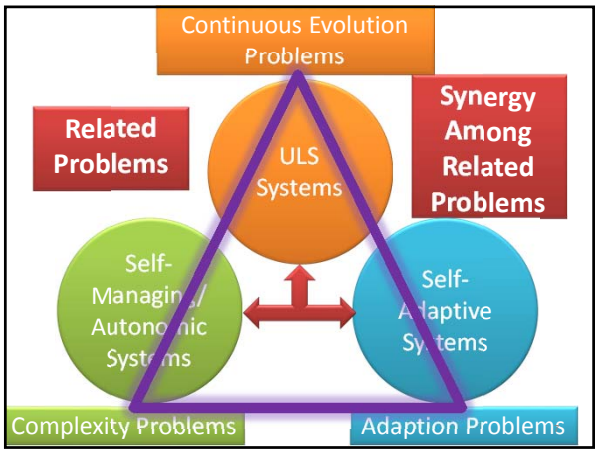
- Study the PID controllers as discussed in class.
- Watch the videos posted in the resource section on PID controllers.
- Define a simple resource control problem.
- Design a simple PID controller for this resource control problem.
- Simulate your PID controller using MATLAB.
- Write a tutorial for software engineering or computer science undergraduate students on how to build a simple PID controller using MATLAB.
  - Introduction: PID Controller Design

The answers for this question should fit into Maximum 4 typeset pages.

You only need to submit one document and m file per group.

Do not copy verbatim from any source. Cite your sources.

Questions
Ron Desmarais
rd@uvic.ca



The Continuous Evolution Problem

Devices, environments, infrastructure, web, services, business goals, user expectations, …

all evolve over time

8

Continuous Evolution

- Traditional (flawed) assumption: software systems should
  - support organizational stability and structure
  - be low maintenance
  - strive for high degrees of user acceptance
- Continuous evolution: software systems
  - should be under constant development
  - can never be fully specified
  - are subject to constant adjustment and adaptation [Truex99]
- Good news
  - for the software engineering community (adaptive, autonomic, reverse engineering in particular) since this view guarantees research problems for years to come
- Bad news
  - most software engineering textbooks will have to be rewritten

Truex et al., Growing Systems in Emergent Organizations, CACM, 1999

Managing Tradeoffs

- From satisfaction of requirements through traditional, top-down engineering

- To satisfaction of requirements by regulation of complex, decentralized systems

How much environment uncertainty can we afford? What's the cost?
What benefits do we accrue by accommodating context uncertainty?

The Complexity Problem

Build a system used by millions of people each day administered and managed by a half-time person

— Jim Gray, Microsoft Research

11

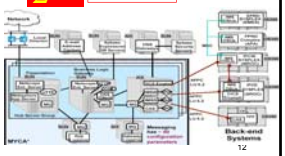
Complexity of Configurations

- Application Server
  - ~100 configuration parameters
  - Several applications
  - Hundreds of servlets
  - Tens of EJBs
- Web Server
  - ~20 configuration parameters
  - Serves thousands of web artifacts
- Messaging
  - ~30 configuration parameters
- DBMS, TCP/IP, OS …

Information systems are very complex for humans and costly to install an maintain

x 2-5 parameters

2^{150} settings



Marin Litoiu, IBM CAS, 2005

12

The Complexity Problem

- The increasing **complexity of computing systems** is overwhelming the capabilities of software developers and system administrators to design, evaluate, integrate, and manage these systems
- Major software and system vendors have concluded that the only viable **long-term solution is to create computing systems that manage themselves**

… an elusive goal?!?

13

The Automation Conundrum

- Over the past 50 years, computer systems have had a huge capacity to automate
  - Enormous variety of tasks
  - Cost per task greatly reduced
  - Incalculable benefits
  - Unprecedented success

- Key challenges
  - Further declines in task costs by traditional methods are subject to the law of **diminishing returns**
  - The **complexity** of infrastructure management threatens to outweigh the benefits of further automation

A. Spector, VP IBM Services and Software Research, 2003

14

Grand Challenge

- Today's computing systems are amazingly complex, and require daunting expertise and patience just to get them running and keep them running
- The increasing system administration will become a major barrier to deploying and maintaining large computing systems

15

Autonomic Computing Vision

Autonomic Computing is really about making systems self-managing …

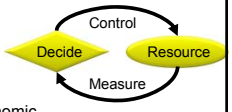
—Paul Horn, IBM Research, 2001

16

What is Autonomic Computing?

- Webster's definition
  - Acting or occurring involuntarily; automatic: an autonomic reflex
  - Relating to, affecting, or controlled by the autonomic nervous system or its effects or activity
  - Autonomic nervous system: that part of the nervous system that governs involuntary body functions like respiration and heart rate

- IBM's definition
  - An approach to self-managed computing systems with a minimum of human interference
  - The term derives from the body's autonomic nervous system, which controls key functions without conscious awareness or involvement



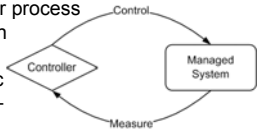
17

Feedback is ubiquitous in natural and engineered systems

18

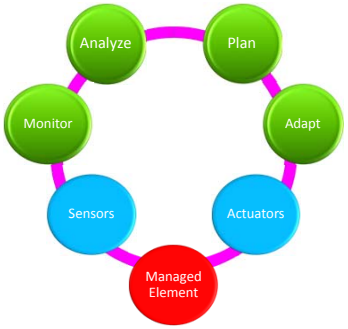
Feedback Systems

- **Merriam-Webster's Online Dictionary**
  the return to the input of a part of the output of a machine, system, or process (as for producing changes in an electronic circuit that improve performance or in an automatic control device that provide self-corrective action)



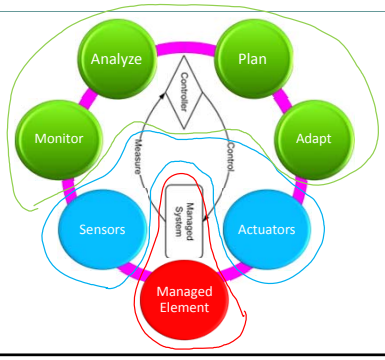
19

Autonomic System



20

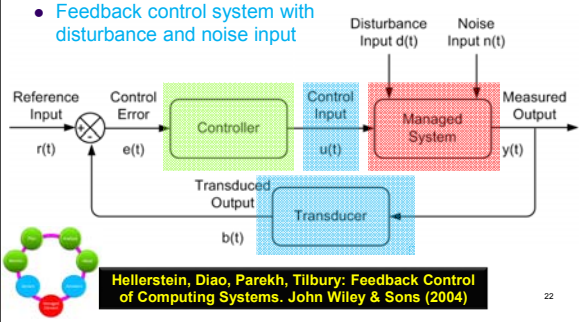
Control Loop Mapping



21

Realization of a Dynamic Architecture

- Feedback control system with disturbance and noise input



Hellerstein, Diao, Parekh, Tilbury: Feedback Control of Computing Systems. John Wiley & Sons (2004)

22

Autonomic Computing Vision

Autonomic Computing is really about making systems self-managing …

—Paul Horn, IBM Research, 2001

23

Reading Assignment

- Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer 36(1):41-50 (2003)
  ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1160055
- IBM: An Architectural Blueprint for Autonomic Computing, 4th Ed. (2006)
  citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.1011&rep=rep1&type=pdf

24

ACRA
AC Reference Architecture

- An Autonomic System (AS) consists of a collection of Autonomic Elements
- An Autonomic Element (AE)
  - Contains resources and delivers services to humans or other autonomic elements
  - Manages its behaviour in accordance with policies that humans or other AEs have established
  - Acts like an agent
    - Autonomous, proactive, goal-directed
    - Interacts with environment

Kephart, Chess: IEEE Computer, 36(1):41-50, Jan. 2003

25

Autonomic Architecture Blueprint

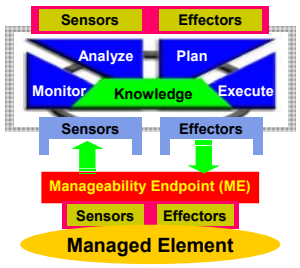
- System building blocks or components
- External interfaces and behaviors of individual building blocks
- Composition and interactions of building blocks so that the individual blocks can contribute toward a common goal
- Composition of blocks into systems so that the system as a whole is self-managing
- Data and control integration standards

IBM: An Architecture Blueprint for Autonomic Computing, 4th Ed. 2006

26

Architectural Building Blocks

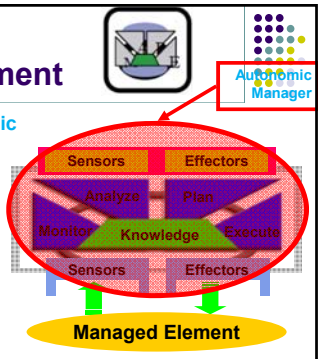
- Autonomic System (AS)
- Autonomic Element (AE)
- Autonomic Manager (AM)
- Managed Element (ME)
- Manageability Endpoint (ME)
- Manageability Interface (MI)
- Knowledge sources
- Enterprise service bus



27

Autonomic Element

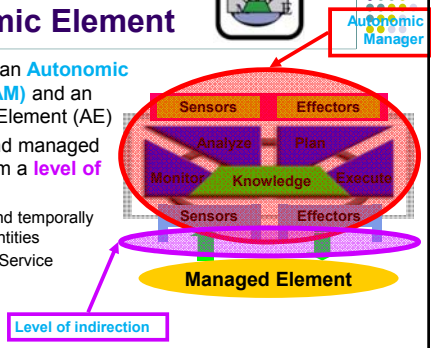
- Consists of an Autonomic Manager (AM) and an Autonomic Element (AE)



28

Autonomic Element

- Consists of an Autonomic Manager (AM) and an Autonomic Element (AE)
- Manager and managed element form a level of indirection
  - Spatially and temporally separate entities
  - Enterprise Service Bus



29

Autonomic Manager

- An Autonomic Manager (AM) automates some management function and externalizes its behavior in its Manageability Interface (MI)
- An AM implements an intelligent control loop
- Self-management—an automated method to
  - Collect information from a resource
  - Filter information and store in a repository
  - Analyze information
  - Create a plan as a sequence of actions
  - Execute those actions

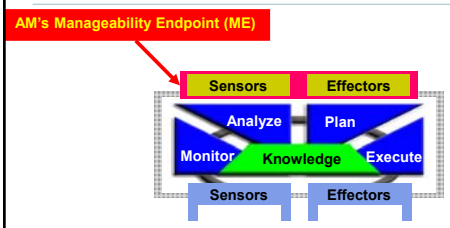
30

Autonomic Manager

- Manages a resource
  - Storage
  - CPU
  - Database
  - Service
  - Legacy system
  - Another Autonomic Element
- Interacts with the managed element through
  - Sensors and effectors
  - Manageability interface
  - Manageability endpoint

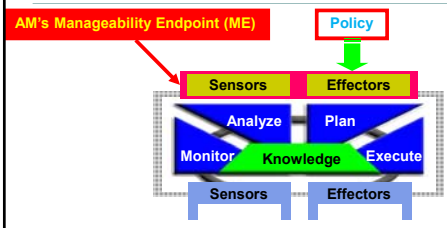
31

Autonomic Manager



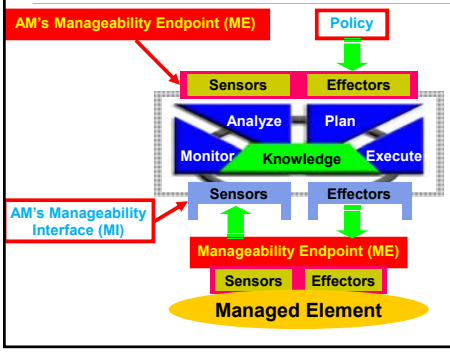
32

Autonomic Manager



33

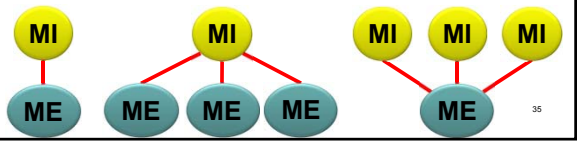
Autonomic Manager



34

Manageability Endpoints

- An autonomic manager communicates with a manageability endpoint through the Manageability Interface (MI)
- A Manageability Endpoint (ME) exposes the state and the management operations for a resource



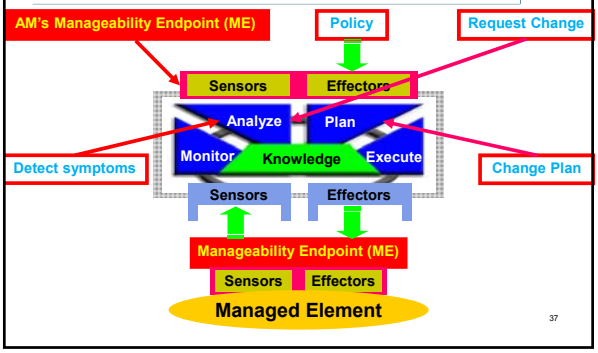
35

Manageability Interface

- An MI for monitoring and controlling a managed resource consists of sensors and effectors
- Sensors obtain data from the resource
  - read state variables in the ME
- Effectors perform operations on the resource
  - call methods in the ME
- Critical success factors for AC initiative
  - Separating AMs and MEs
  - Standardizing MIs

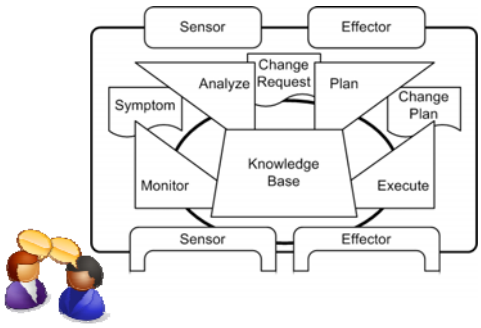
36

Autonomic Manager



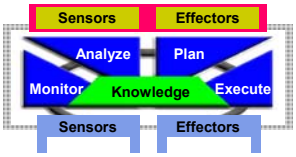
37

Autonomic Manager



38

Autonomic Manager
MAPE Loop or MAPE-K Loop



39

MAPE-K Loop
Monitor

- Senses the managed process and its context
- Collects data from the managed resource
- Provides mechanisms to aggregate and filter incoming data stream
- Stores relevant and critical data in the knowledge base or repository for future reference.

Analyzer

- Compares event data against patterns in the knowledge base to diagnose symptoms and stores the symptoms
- Correlates incoming data with historical data and policies stored in repository
- Analyzes symptoms
- Predicts problems

40

MAPE-K Loop
Planner

- Interprets the symptoms and devises a plan
- Decides on a plan of action
- Constructs actions
  - building scripts
- Implements policies
- Often performed manually

Execute Engine

- Executes the change in the managed process through the effectors
- Perform the execution plan
- Often performed manually

41

MAPE-K Loop
Knowledge Base

- The four components of a MAPE-K loop work together by exchanging knowledge through the knowledge base to achieve the control objective.
- An autonomic manager
  - maintains its own knowledge
    - Information about its current state as well as past states
  - But also has access to knowledge which is shared among collaborating autonomic managers
    - Configuration database, symptoms database, business rules, provisioning policies, or problem determination expertise

42