

Fault Supervision for Multi Robotics Systems

Felipe de Fraga Roman

Advisor: Prof. Dr. Alexandre de Morais
Amory



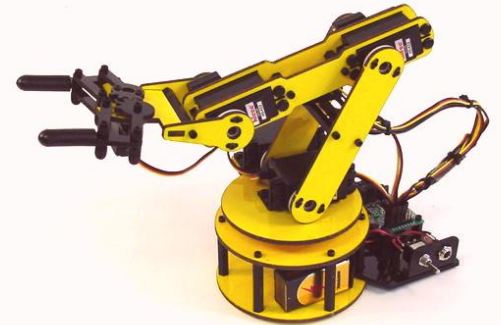
Agenda

- Introduction
- Motivations and Goals
- Theoretical Background
- State of the Art
- Research Proposal



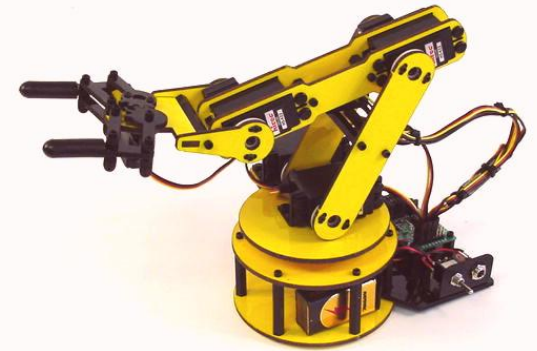
Introduction

- Robotics becomes more common
- Variety of tasks
 - Robot arm
 - Automated guided vehicles
 - Unmanned aerial vehicles
 - Humanoid robots
- Others...



Introduction

- Two different kinds of robots
 - Stationary robots
 - Mobile robots



Introduction

- Mobile robotics is the research area that handles the control of autonomous vehicle or semi-autonomous vehicle.
- Currently, there are few commercial applications of mobile service robots
 - Robotics has been evolving fast in terms of new functionalities and becoming affordable



Introduction

- Mobile robots have not yet made much impact upon industrial and domestic applications, mainly due to the lack of dependability, robustness, reliability and flexibility in real environments.



Introduction

- One cost-effective way to provide effectiveness and robustness to robotic system is to use multi-robots instead of a single robot
- MRS have some advantages over single-robots systems
 - *Increased of speed*
 - *Task completion through parallelism*
 - *Increased of robustness and reliability*



Introduction

Classification of MRS

- Homogeneous
 - All members of the team have the same specification
- Heterogeneous
 - Different kind of robots in the same team



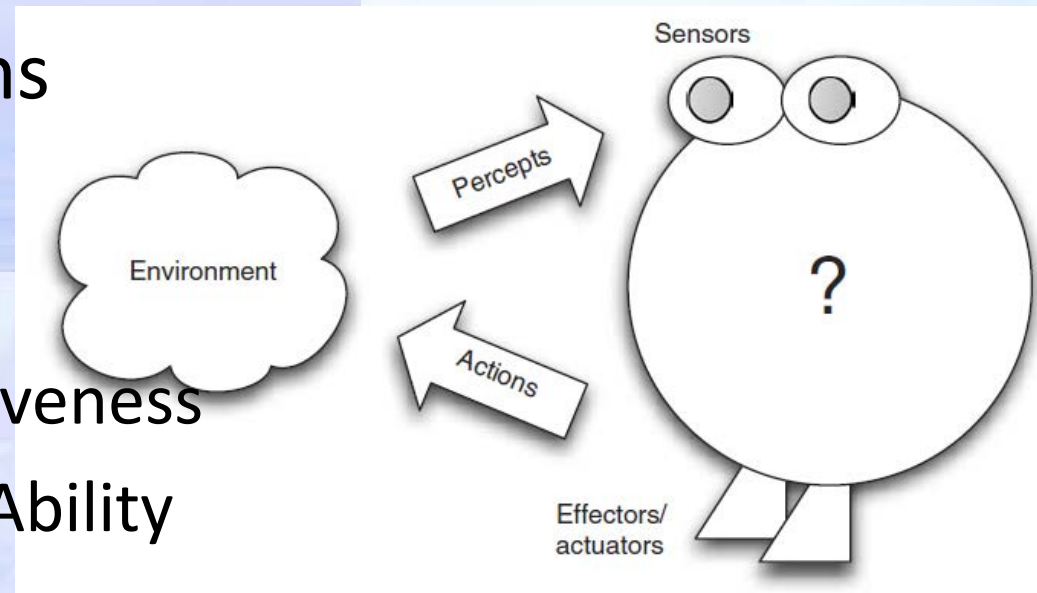
Motivations and Goals

- Mobile robotics will become commonplace, cost-effective and dependable
- The goal of this work is to provide monitor faults at a team of heterogeneous robotic agents
- More dependable, More applications



Theoretical Background

- Autonomous Agents
- Characteristics of Agents
 - Autonomy
 - Pro-activeness
 - Re-activeness
 - Social Ability



Theoretical Background

- The ***dependability*** of a computing system is its ability to deliver service that can be trusted;
- ***Correct service*** is delivered when the service implements the system *function*, that is what the system is intended to do.



Figure 1: Dependability tree

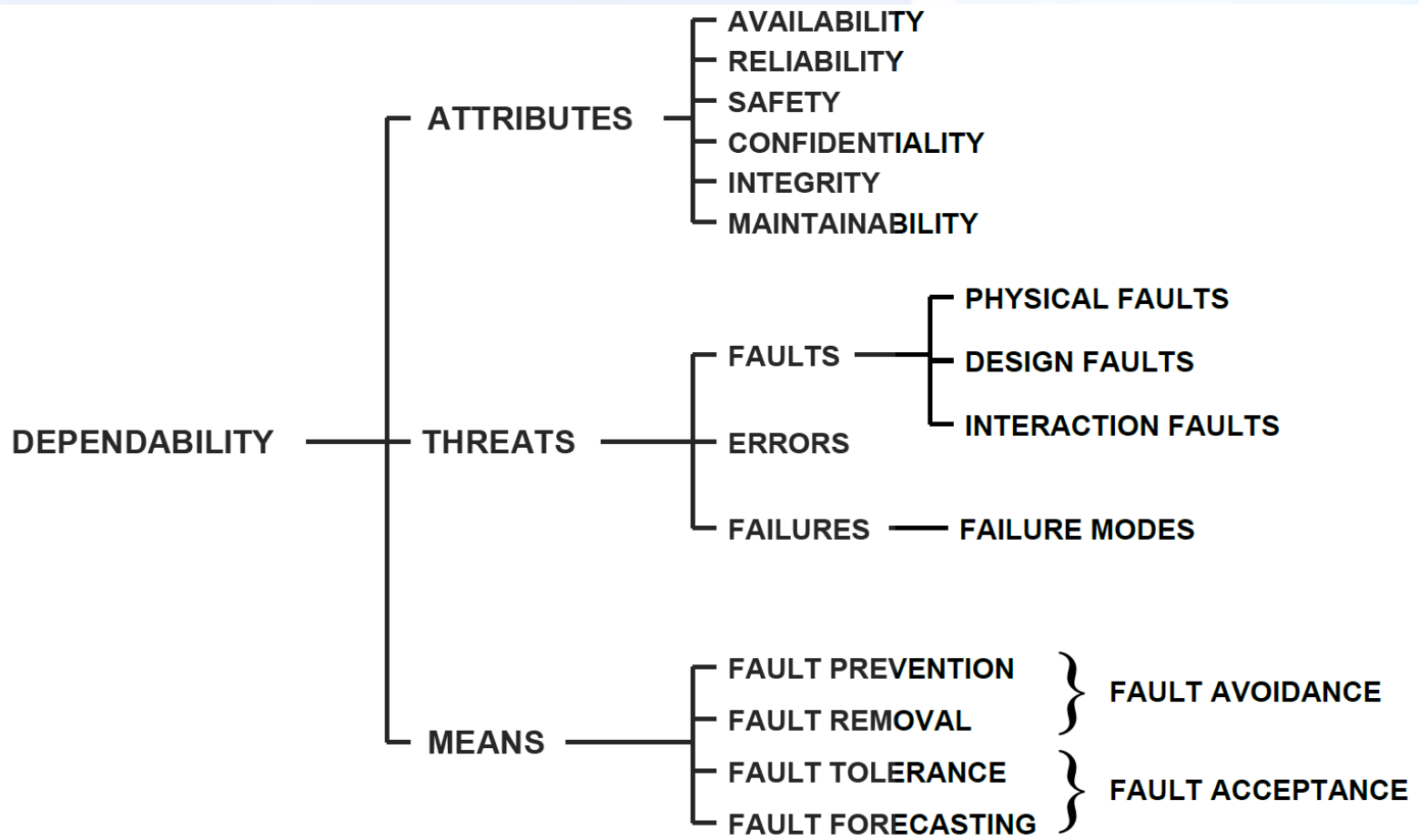
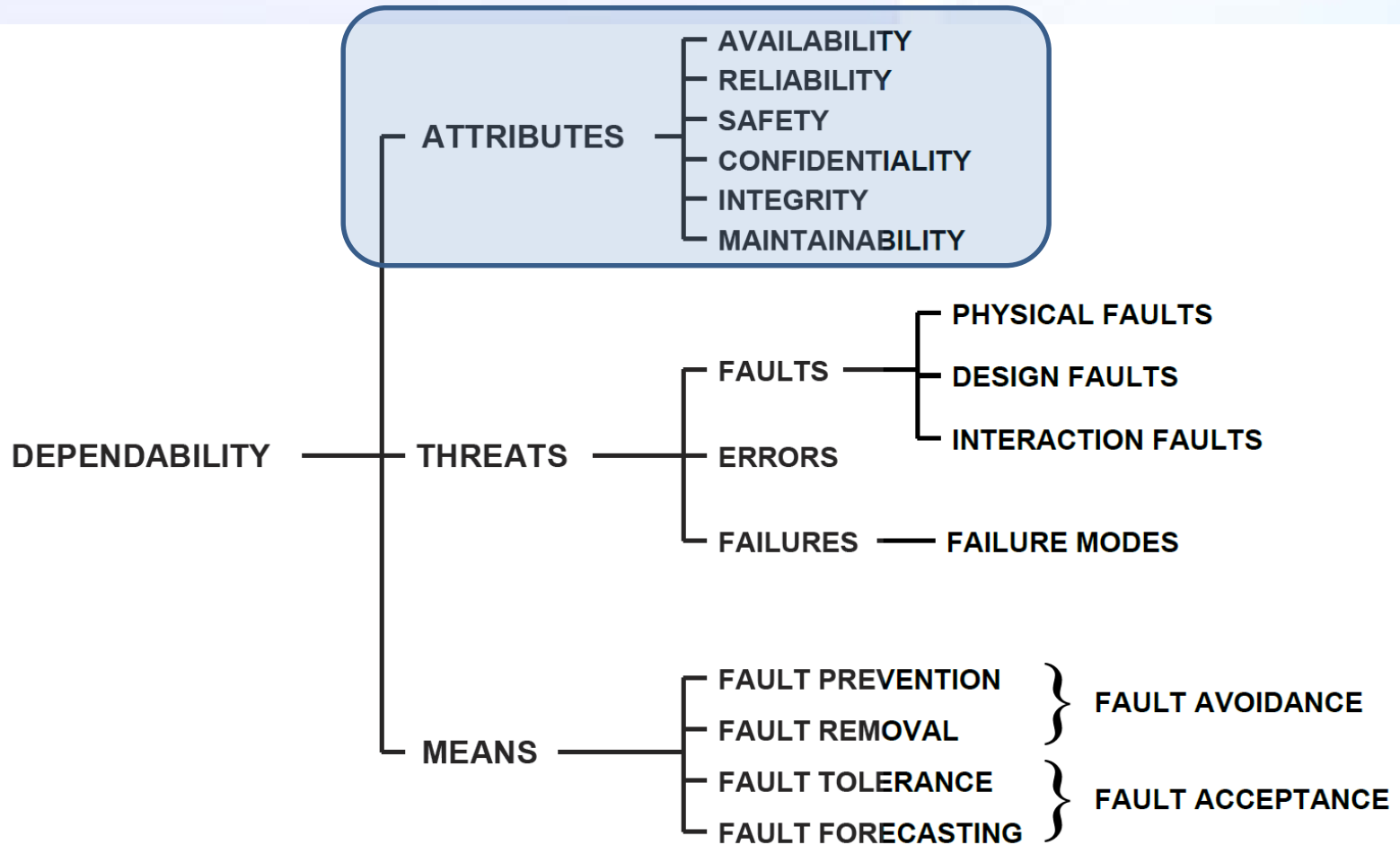


Figure 1: Dependability tree



Basic concepts

- ***Availability***: the deliverance of correct service at a given time/period of time,
- ***Reliability***: the continuous deliverance of correct service for a period of time,
 - ***Safety***: the absence of catastrophic consequences on the users and the environment,

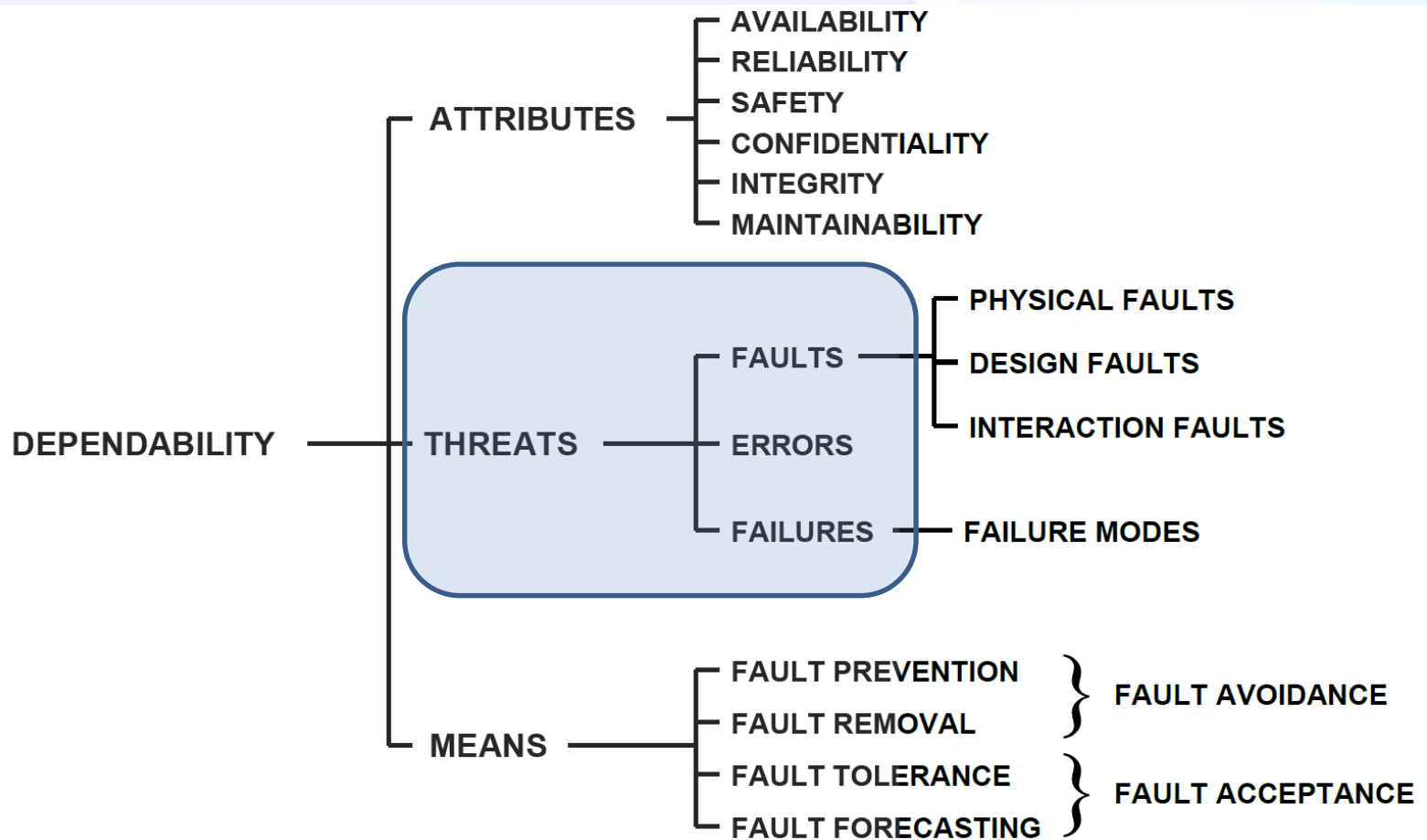


Basic concepts

- ***Confidentiality***: the absence of unauthorized disclosure of information,
- ***Integrity***: the absence of improper system state alterations,
- ***Maintainability***: the ability to do repairs and modifications



Figure 1: Dependability tree



Basic concepts

- The ***threats*** to a system's dependability consist of **failures, errors and faults**
- A system ***failure*** is an event that occurs when the delivered service deviates from *correct service*.
- An ***error*** is that part of the system state that can cause a subsequent failure.



Basic concepts

- A **fault** is the cause of an error.

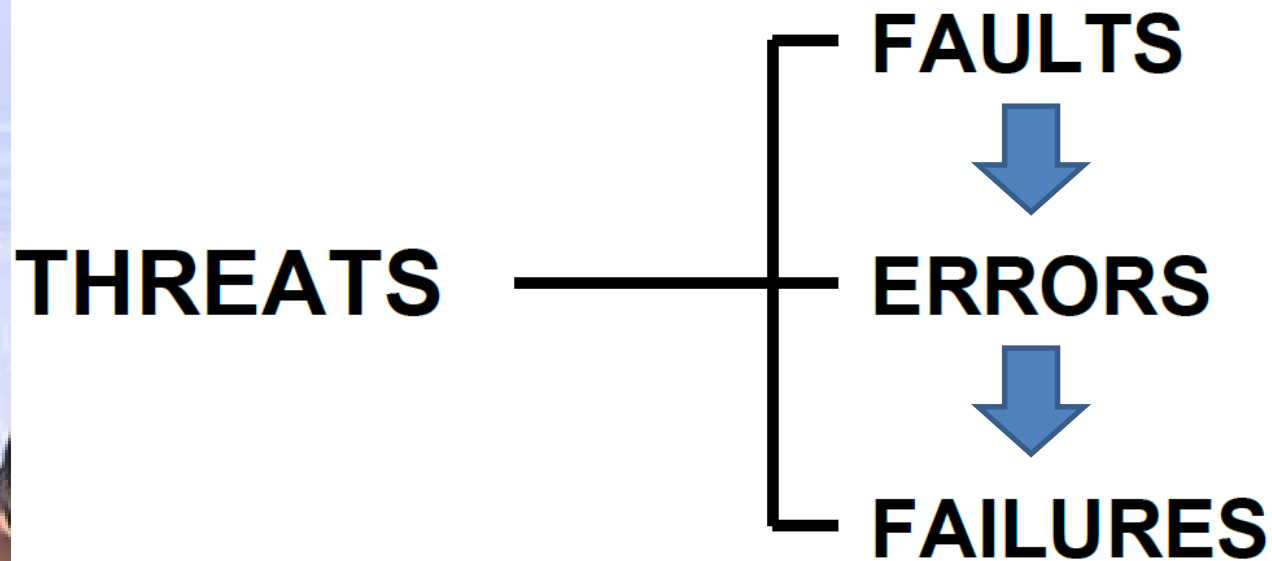
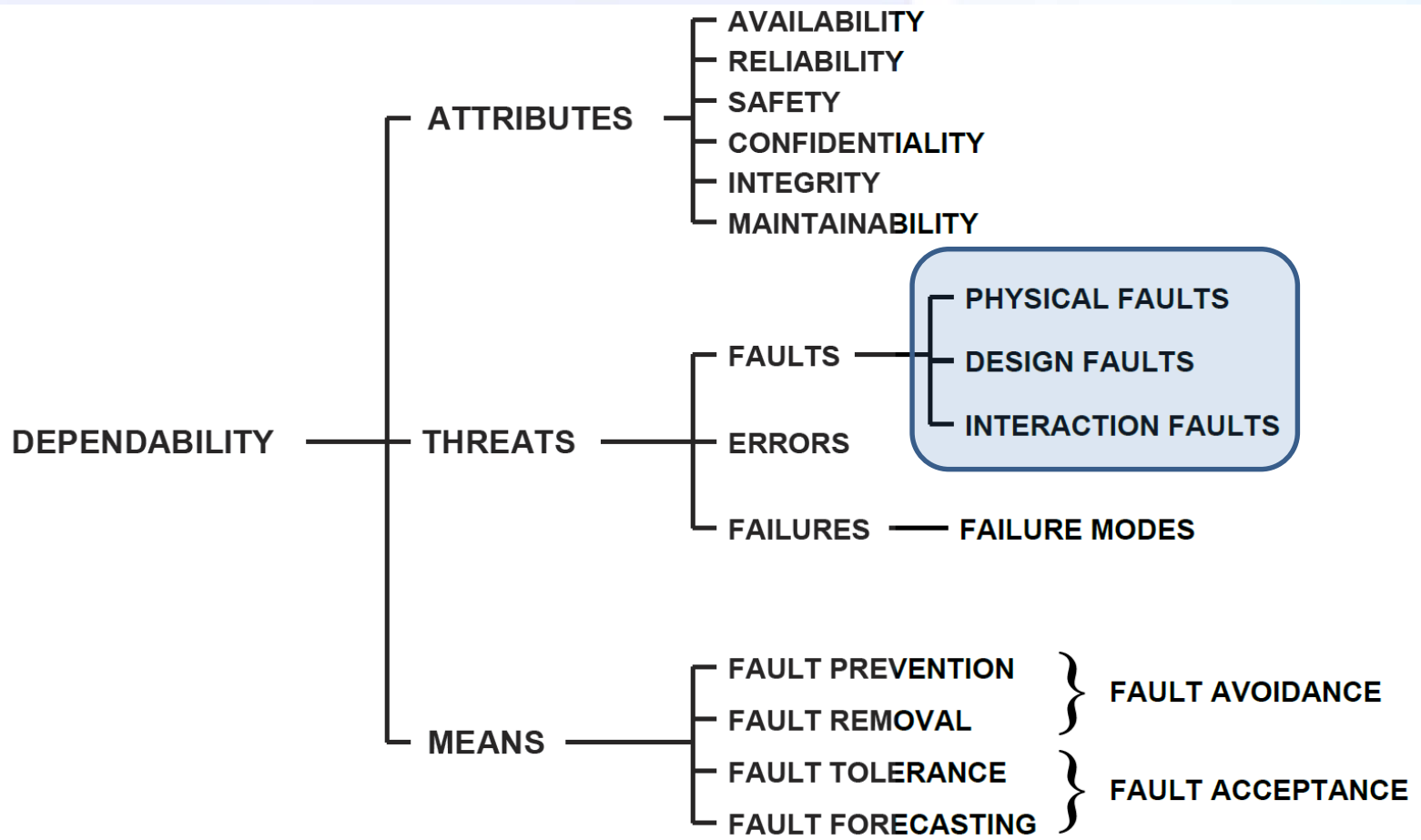


Figure 1: Dependability tree

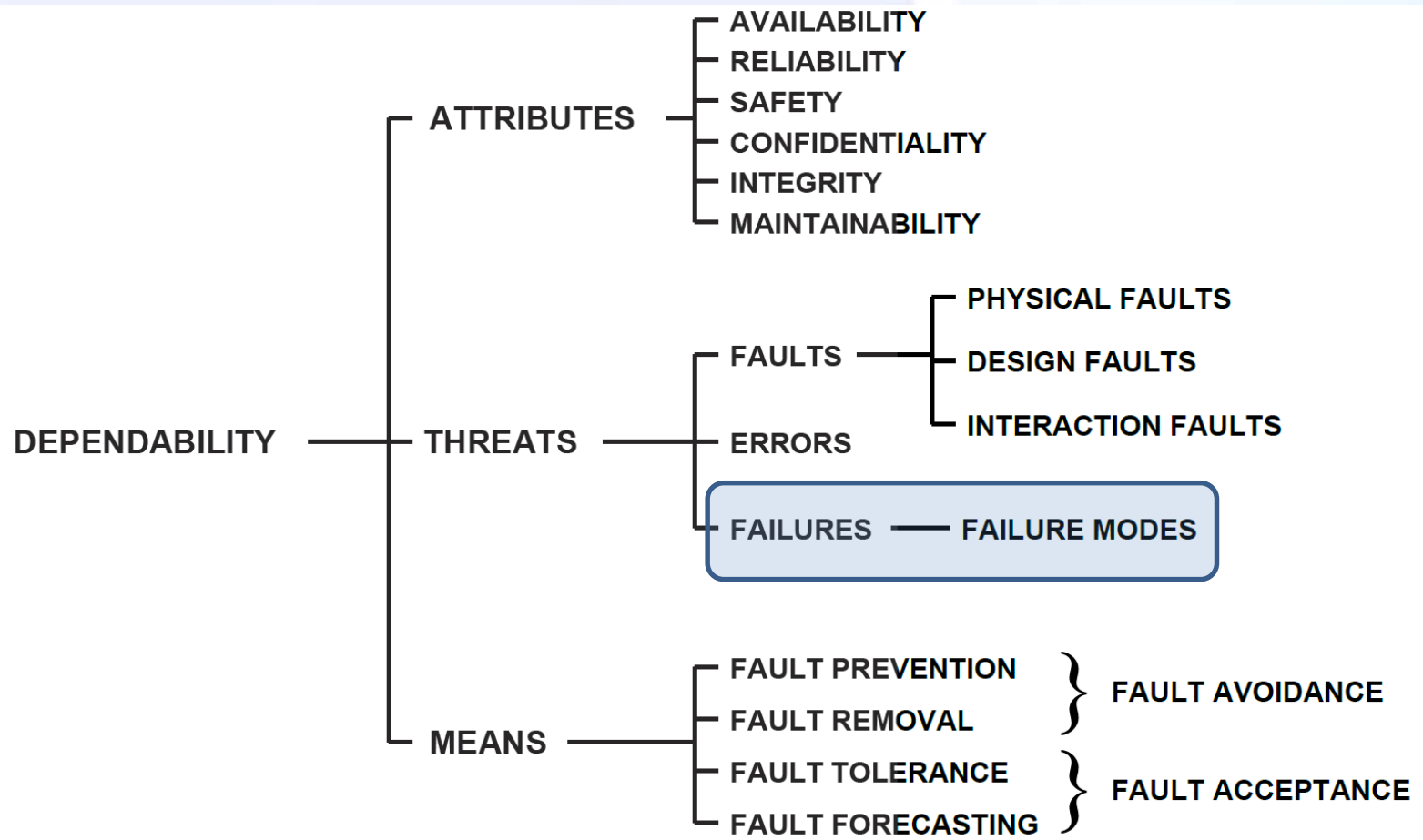


Basic concepts

- ***physical faults*** are faults due to adverse physical phenomena,
- ***design faults*** are faults unintentionally caused by man during the development of the system,
- ***interaction faults*** are faults resulting from the interaction with other systems, including users



Figure 1: Dependability tree

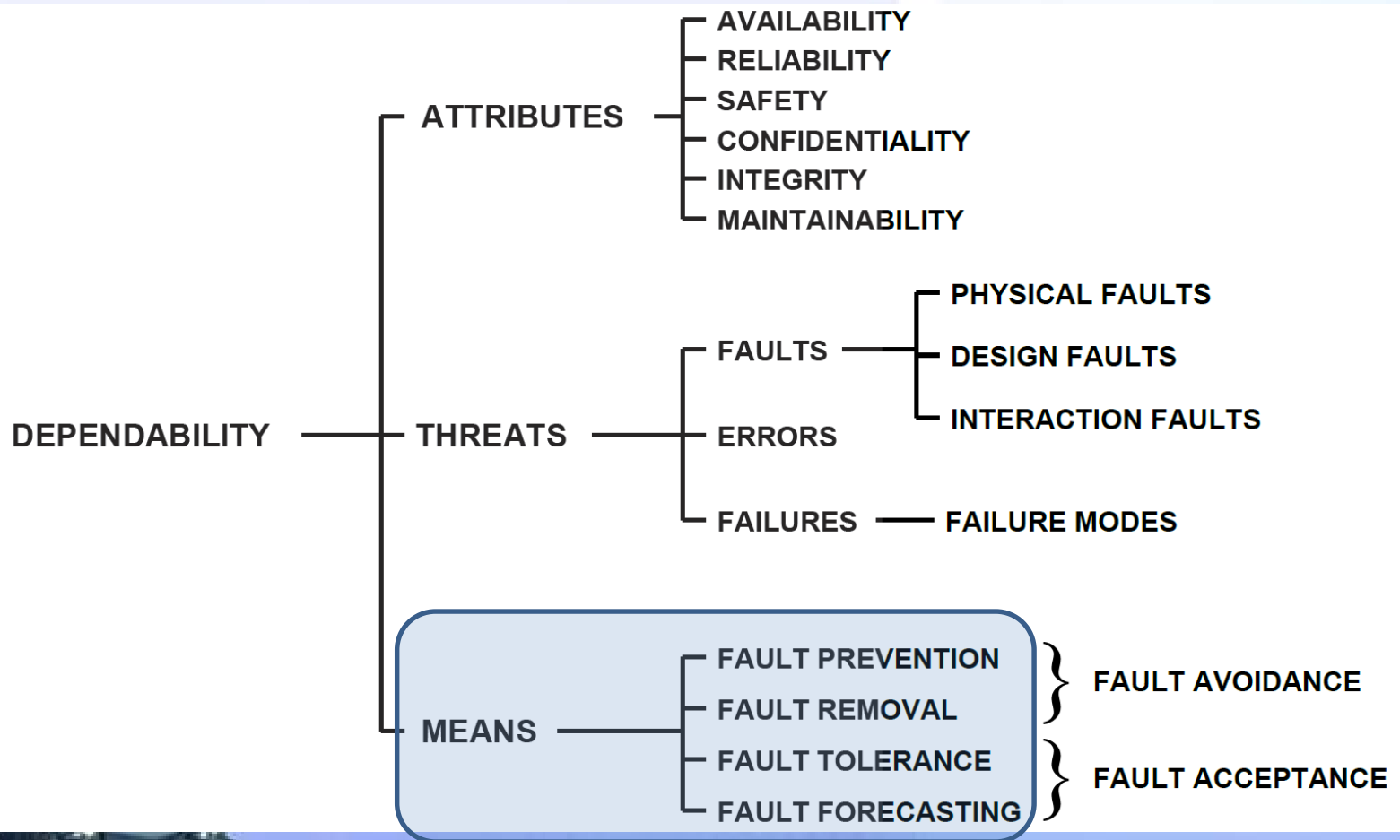


Basic concepts

- The way in which a system can fail are its *failure modes*, characterized by the severity and the symptoms of a failure.



Figure 1: Dependability tree



Basic concepts

- ***fault prevention***: how to prevent the occurrence or introduction of faults,
- ***fault removal***: how to reduce the number or severity of faults,

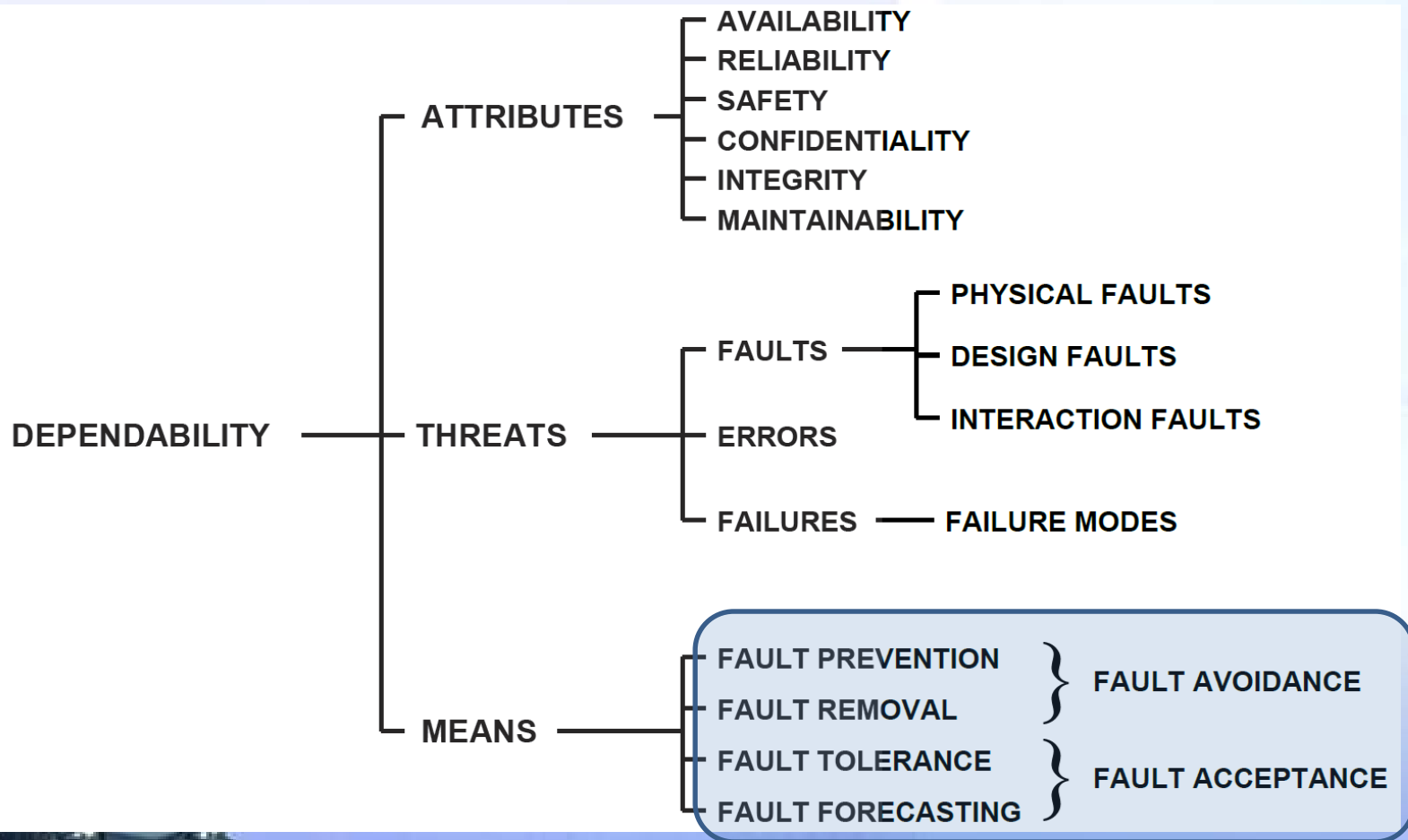


Basic concepts

- ***fault tolerance***: how to deliver *correct service* in the presence of faults,
- ***fault forecasting***: how to estimate the present number, the future incidence, and the likely consequences of faults



Figure 1: Dependability tree



Fault tolerance

- Fault tolerance is intended to preserve the delivery of correct service in the presence of active faults
- Fault tolerance mechanisms:
 - ***Recovery*** transforms a system that contains errors into a state without detected errors



Fault tolerance - Recovery

- *Rollback,*
- *Rollforward,*
- *Compensation;*

** There are fault tolerance mechanisms for each find of faults*



Theoretical Background

Reliability in Multiple Robotics Systems

- MRS need to be reliable as a whole
- Questions to be addressed:
 - How to detect when robots have failed?
 - How to diagnose robots failures?
 - How to respond to these failures?



Theoretical Background

- Challenges of achieve reliability in MRS:
 - Individual robot failure
 - Local perspective
 - Interference
 - Software errors
- Communication failures



State of the Art

- There are large possibilities of faults in robotics:
 - Robot sensors faults
 - Uncertain environment models
 - Limited power and computation limits



State of the Art

- Robot middlewares try to address the fault detection problem
- Only single parts of the problem are addressed
 - Each one of these middleware monitoring tools starts from scratch
- Most of them are driven by the capabilities of the robotics middleware and not by the robotics field needs



State of the Art

Individual Robots Fault Detection

- Thresholds: comparing the sensors values with a pre-determined range of acceptable values
- Vote system: based on different redundant components
- Off-line fault detection: Logging is technique where data is collected in advance to be analyzed later



State of the Art

Multiple Robots Fault Detection

- Fault detection systems in MRS have the distribution as a coefficient that increases the complexity
- The MRS must be able to cooperate and communicate with each other
- A networked control system is a requirement to connect all agents through communication



State of the Art

Multiple Robots Fault Detection

- There are several methods and techniques to deal with
- Centralized designed
 - without attending the distributed and decentralized nature



State of the Art

Distributed Artificial Intelligence

- Creation of a supervision system agent
- Able to communicate with other
- Perform monitor tasks



State of the Art

Swarm robotic systems

- Advantage is the redundancy
- Another robot can take steps to repair
- Take over the failed robot's task



State of the Art

Monitoring by Flashes

- Each robot flashes by lighting up its on-board light-emitting diodes
- Neighboring robots are driven to flash in synchrony
- Error robots do not flash periodically



Research Proposal

Research Problem

Even MRS designed to be robust will face unexpected faults from a very large range of possibilities

Goals

The goal of this work is to propose a fault monitoring tool for MRS

- Integrate a infrastructure networking monitoring tool with a robotics middleware



Research Proposal

- **Research Questions**
- Is it possible to adapt an IT infrastructure monitoring tool to detecting faults in MRS?
- How effective this monitoring system will be?



Research Proposal

Robots middleware

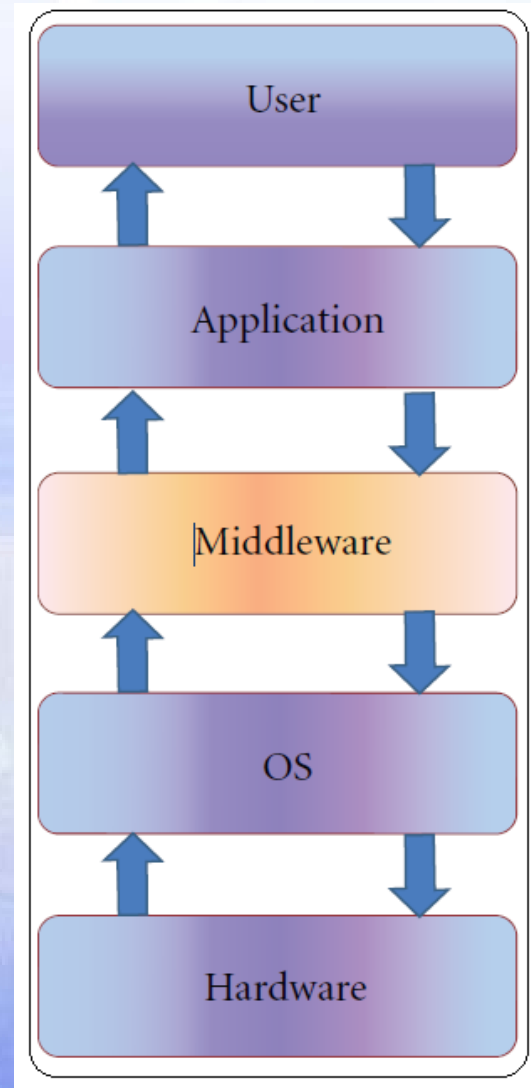
- Robot Operating System (ROS) is a middleware that provides a communication layer above the host operating system of a heterogeneous computing node



Research Proposal

Robots middleware

ROS



Research Proposal

IT Monitoring Tool

- Nagios provides information about mission-critical IT infrastructure, allowing detecting and repairing problems and mitigating future issues

Nagios supports plugins/extensions



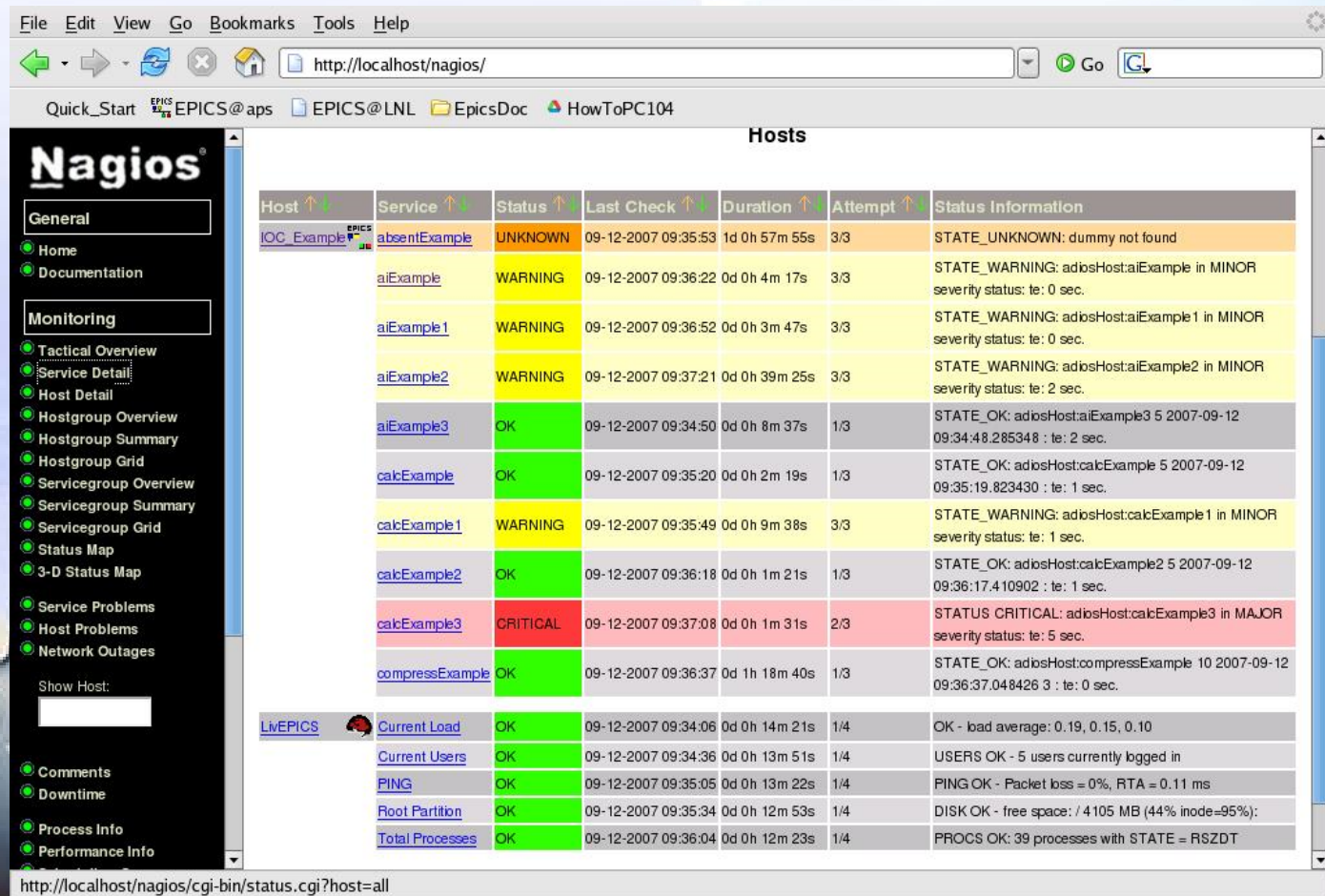
Research Proposal

Nagios Plugins

- These plugins can monitor virtually any kind of equipment/devices
- The proposal is to develop a custom plugin to monitor both software information and also hardware information



Nagios screenshots



The screenshot shows the Nagios web interface in a browser window. The browser's address bar displays 'http://localhost/nagios/'. The interface includes a sidebar with navigation links under 'General' and 'Monitoring'. The main content area, titled 'Hosts', displays a table of monitored hosts and their services. The table columns are: Host, Service, Status, Last Check, Duration, Attempt, and Status Information. The table lists various hosts like 'IOC_Example', 'aiExample', 'aiExample1', 'aiExample2', 'aiExample3', 'cacExample', 'cacExample1', 'cacExample2', 'cacExample3', 'compressExample', and 'LivePICS'. Each host has multiple services listed, such as 'absentExample', 'aiExample', 'aiExample1', 'aiExample2', 'aiExample3', 'cacExample', 'cacExample1', 'cacExample2', 'cacExample3', and 'compressExample'. The status of each service is indicated by a colored box: UNKNOWN (orange), WARNING (yellow), OK (green), and CRITICAL (red). The status information column provides details about the service's state, including severity status and time to event (te).

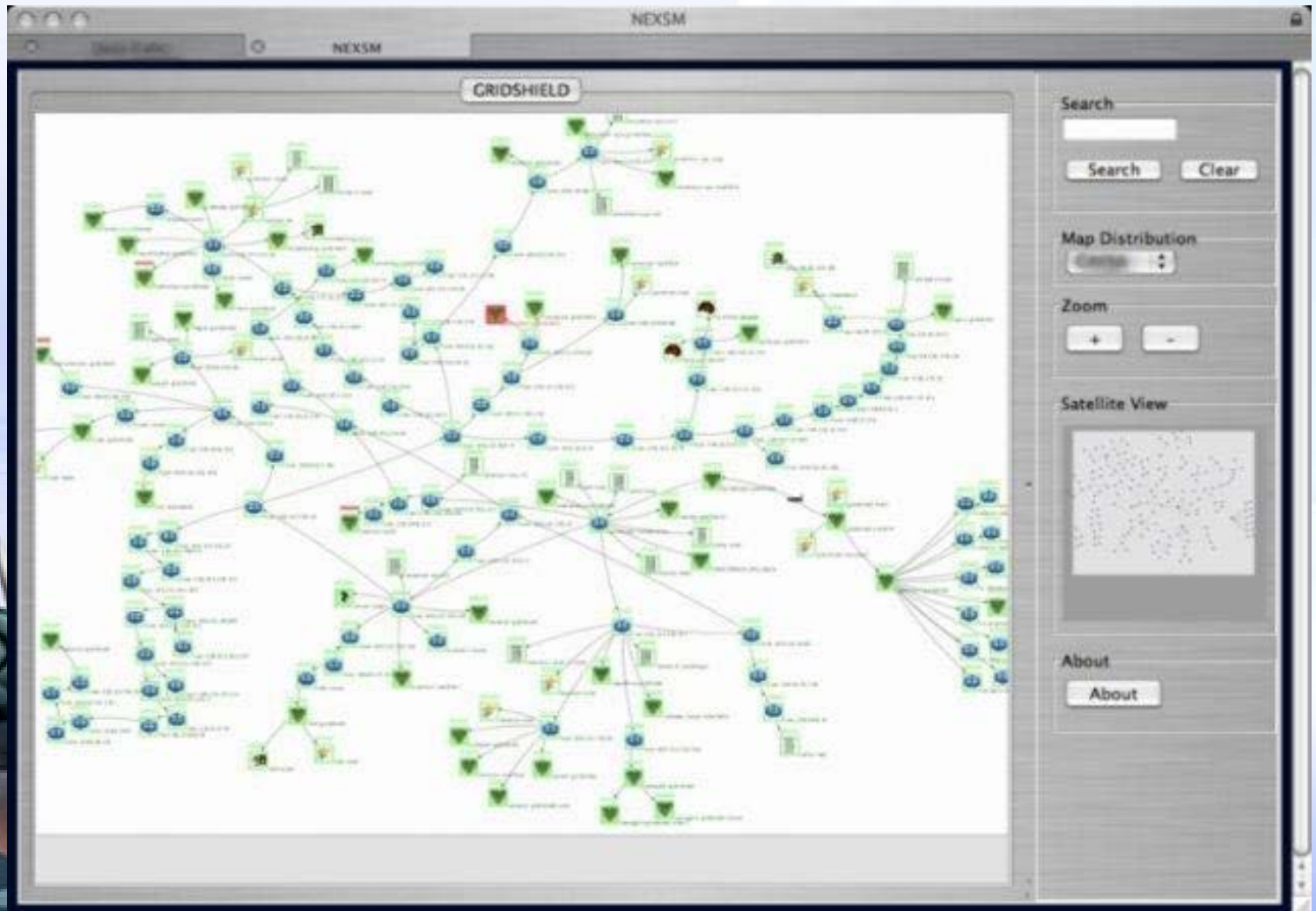
Host	Service	Status	Last Check	Duration	Attempt	Status Information
IOC_Example	absentExample	UNKNOWN	09-12-2007 09:35:53	1d 0h 57m 55s	3/3	STATE_UNKNOWN: dummy not found
	aiExample	WARNING	09-12-2007 09:36:22	0d 0h 4m 17s	3/3	STATE_WARNING: adiosHost:aiExample in MINOR severity status: te: 0 sec.
	aiExample1	WARNING	09-12-2007 09:36:52	0d 0h 3m 47s	3/3	STATE_WARNING: adiosHost:aiExample1 in MINOR severity status: te: 0 sec.
	aiExample2	WARNING	09-12-2007 09:37:21	0d 0h 39m 25s	3/3	STATE_WARNING: adiosHost:aiExample2 in MINOR severity status: te: 2 sec.
	aiExample3	OK	09-12-2007 09:34:50	0d 0h 8m 37s	1/3	STATE_OK: adiosHost:aiExample3 5 2007-09-12 09:34:48.285348 : te: 2 sec.
	cacExample	OK	09-12-2007 09:35:20	0d 0h 2m 19s	1/3	STATE_OK: adiosHost:cacExample 5 2007-09-12 09:35:19.823430 : te: 1 sec.
	cacExample1	WARNING	09-12-2007 09:35:49	0d 0h 9m 38s	3/3	STATE_WARNING: adiosHost:cacExample1 in MINOR severity status: te: 1 sec.
	cacExample2	OK	09-12-2007 09:36:18	0d 0h 1m 21s	1/3	STATE_OK: adiosHost:cacExample2 5 2007-09-12 09:36:17.410902 : te: 1 sec.
	cacExample3	CRITICAL	09-12-2007 09:37:08	0d 0h 1m 31s	2/3	STATUS CRITICAL: adiosHost:cacExample3 in MAJOR severity status: te: 5 sec.
	compressExample	OK	09-12-2007 09:36:37	0d 1h 18m 40s	1/3	STATE_OK: adiosHost:compressExample 10 2007-09-12 09:36:37.048426 3 : te: 0 sec.
LivePICS	Current Load	OK	09-12-2007 09:34:06	0d 0h 14m 21s	1/4	OK - load average: 0.19, 0.15, 0.10
	Current Users	OK	09-12-2007 09:34:36	0d 0h 13m 51s	1/4	USERS OK - 5 users currently logged in
	PING	OK	09-12-2007 09:35:05	0d 0h 13m 22s	1/4	PING OK - Packet loss = 0%, RTA = 0.11 ms
	Root Partition	OK	09-12-2007 09:35:34	0d 0h 12m 53s	1/4	DISK OK - free space: / 4105 MB (44% inode=95%):
	Total Processes	OK	09-12-2007 09:36:04	0d 0h 12m 23s	1/4	PROCS OK: 39 processes with STATE = RSZDT



Nagios screenshots



Nagios screenshots



QA



Thank you

