

A Survey on Simulation-Based Fault Injection Tools for Complex Systems

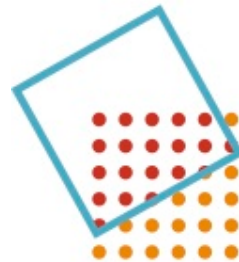
SETS 2014

Maha Kooli

PhD Student in LIRMM

Giorgio Di Natale, Alberto Bosio

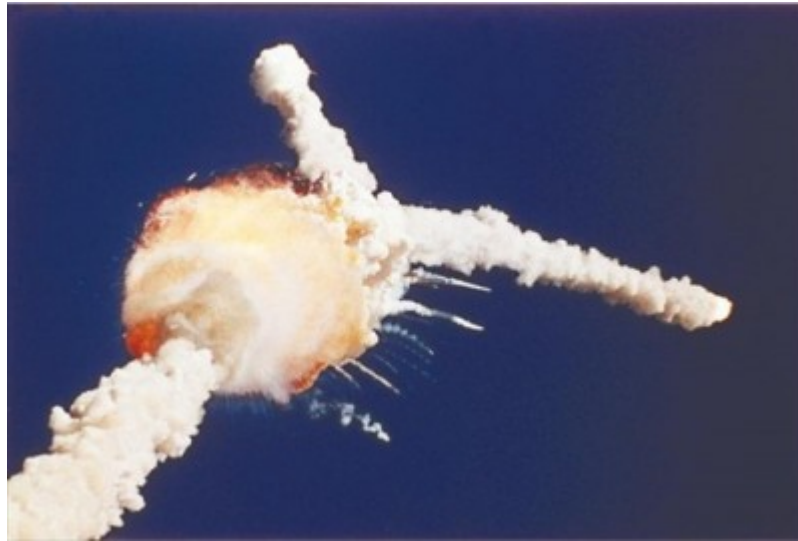
Pascal Benoit, Lionel Torres



Laboratoire
Informatique
Robotique
Microélectronique
Montpellier



Motivation



Space shuttle shatter the axis of NASA
January 28, 1986

Outline

- 1. CLERECO Project**
- 2. State of the Art**
 - 2.1. Dependability**
 - 2.2. Fault Tolerance**
 - 2.3. Fault Injection**
- 3. Research Direction**
- 4. Conclusion and Perspective**

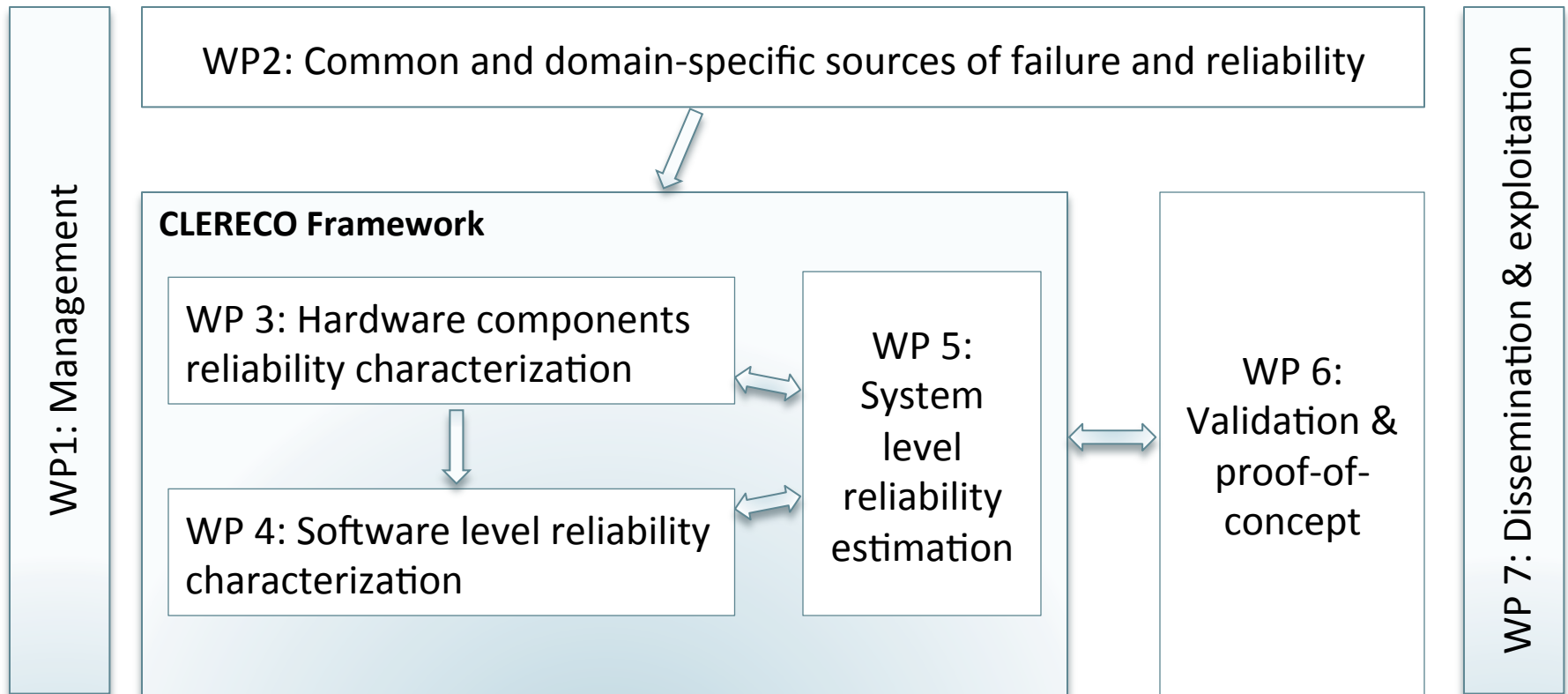
Outline

- 1. CLERECO Project**
2. State of the Art
 - 2.1. Dependability
 - 2.2. Fault Tolerance
 - 2.3. Fault Injection
3. Research Direction
4. Conclusion and Perspective

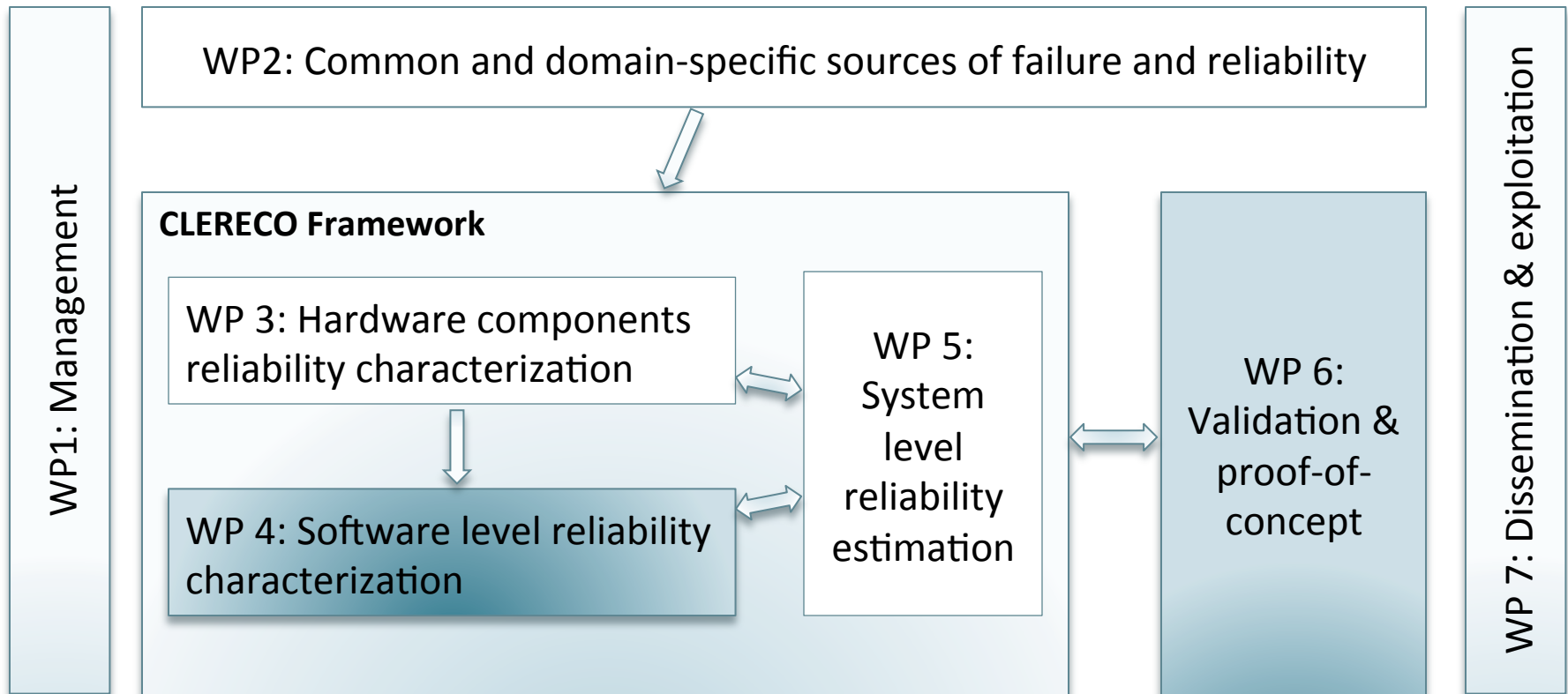
CLERECO



CLERECO work packages



CLERECO work packages



Outline

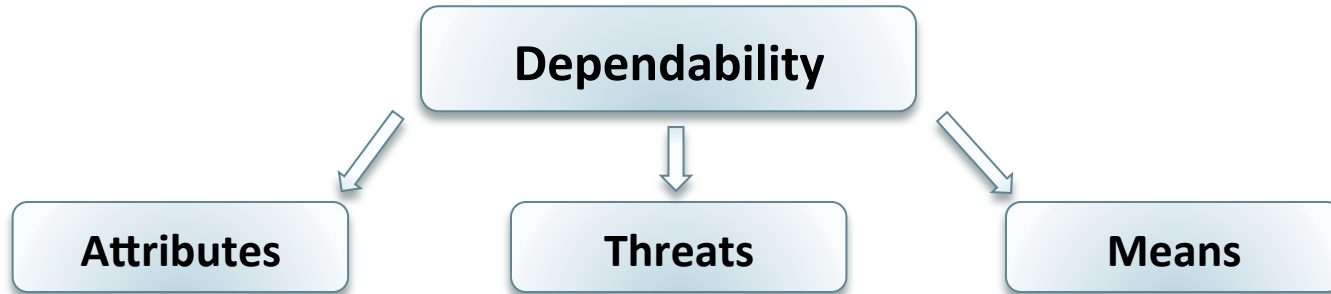
1. CLERECO Project
2. **State of the Art**
 - 2.1. **Dependability**
 - 2.2. Fault Tolerance
 - 2.3. Fault Injection
3. Research Direction
4. Conclusion and Perspective

Dependability

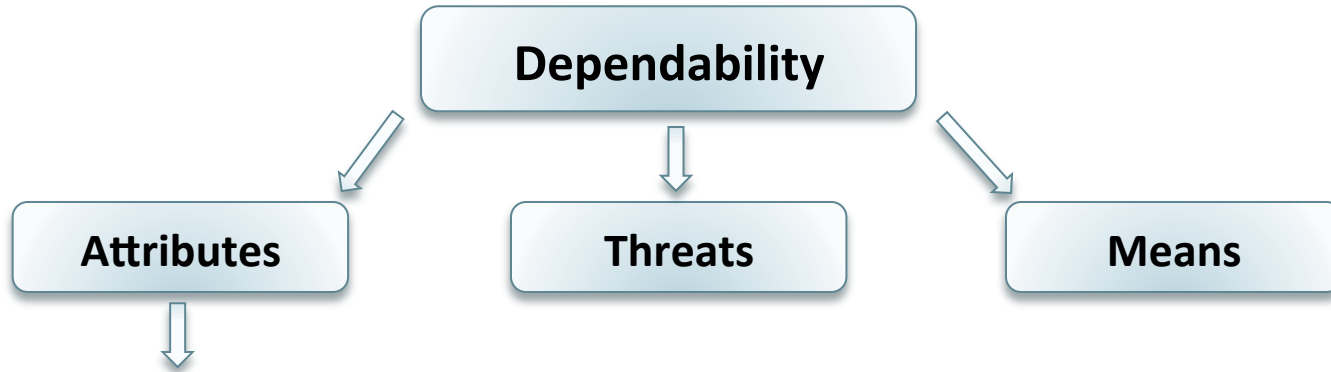
Dependability

- Dependability is a global concept that subsumes the usual attributes of reliability, availability, safety, integrity, and maintainability.
- Dependability represents the ability to avoid service failures that can happen to a system frequently and severely than acceptable.

Dependability

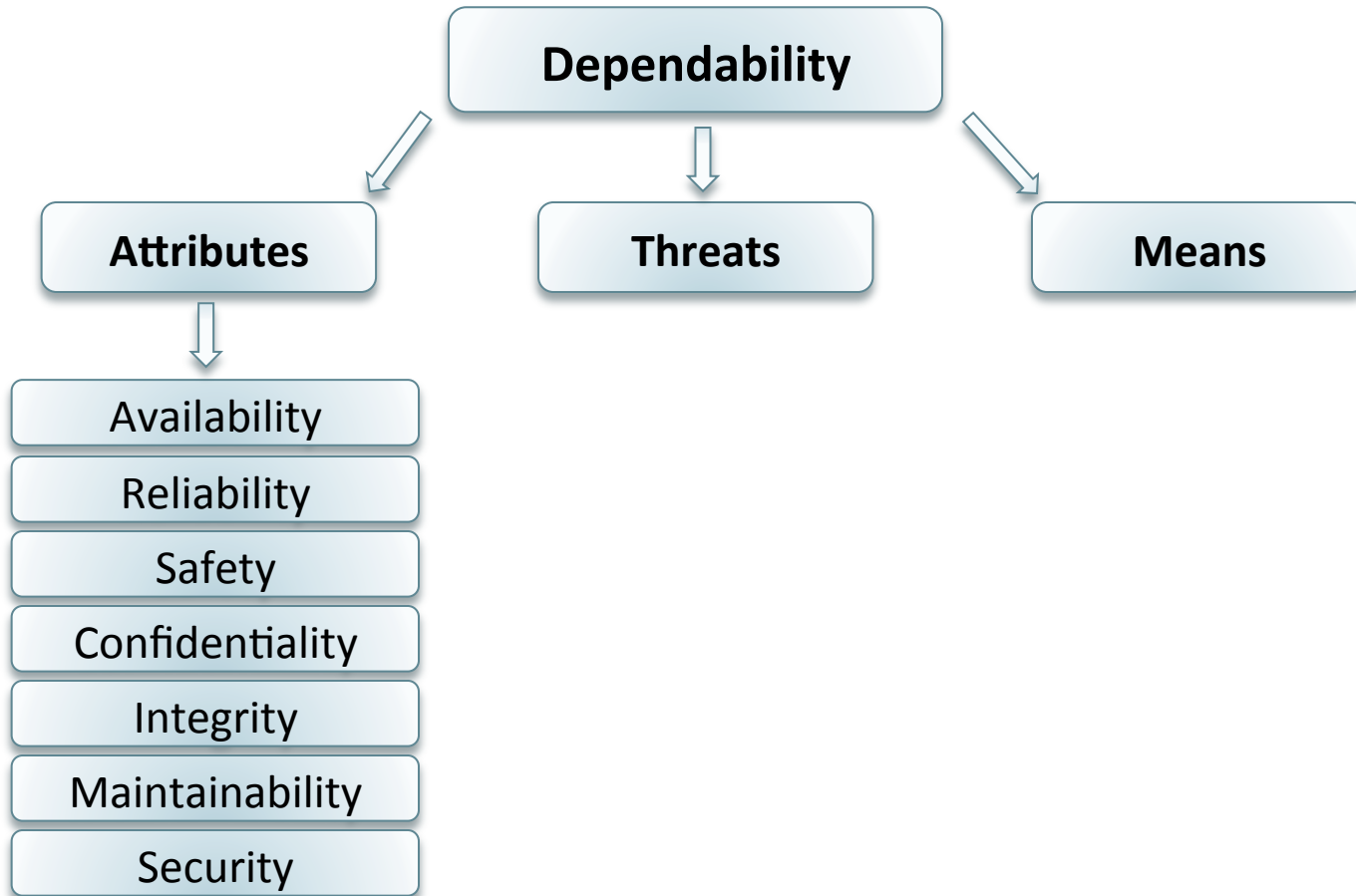


Dependability



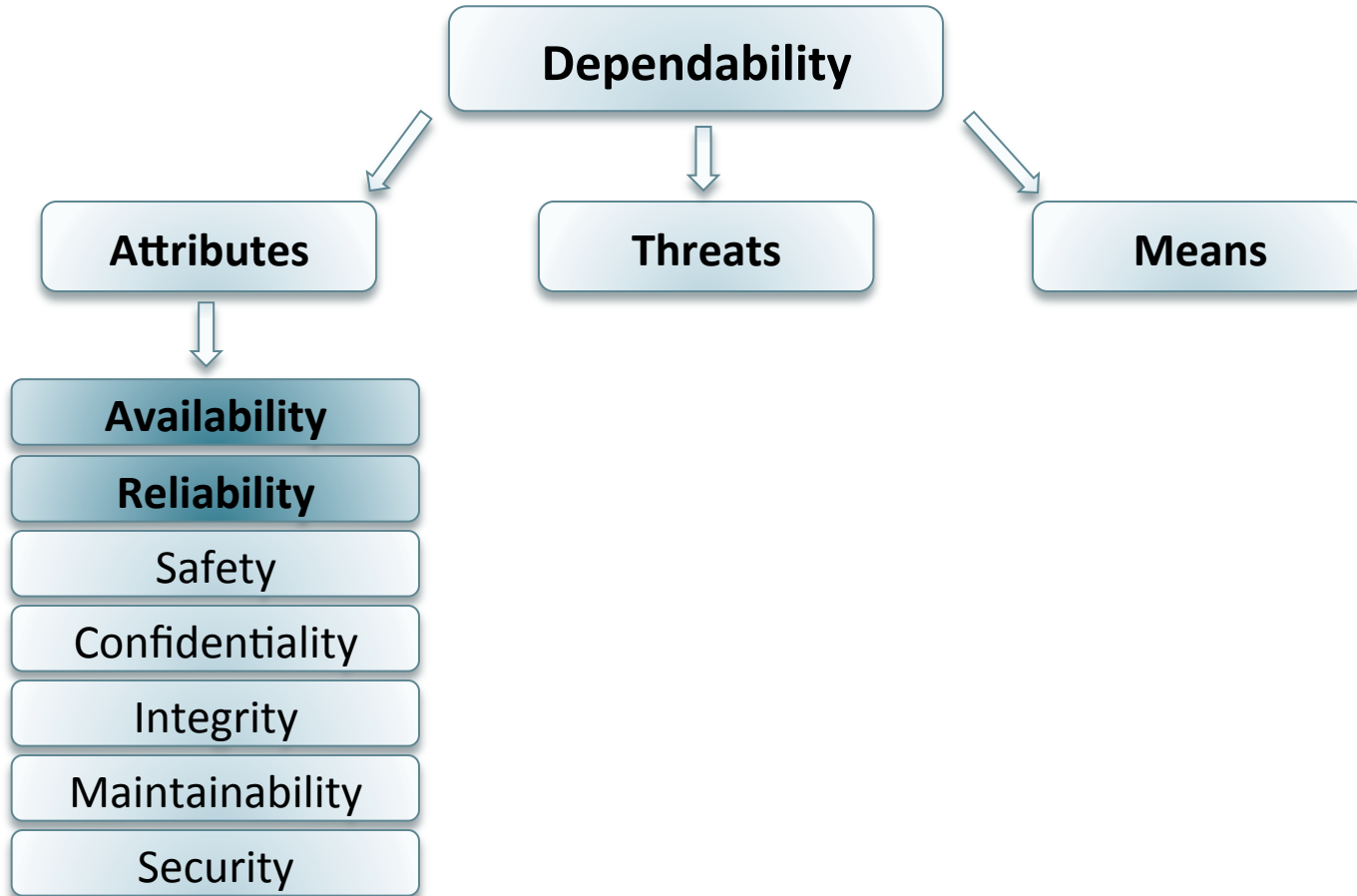
- A way to evaluate the dependability.

Dependability



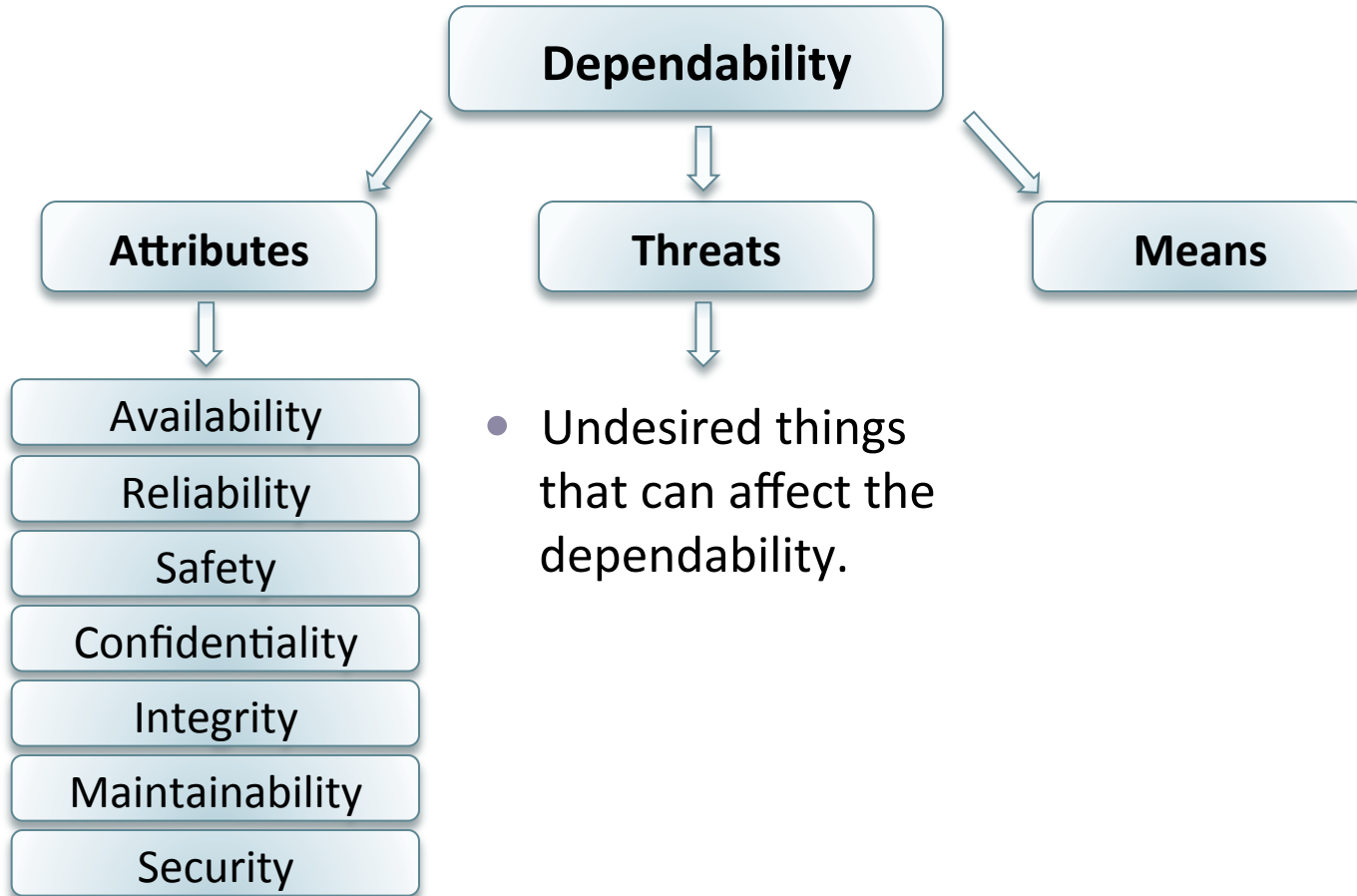
etc...

Dependability



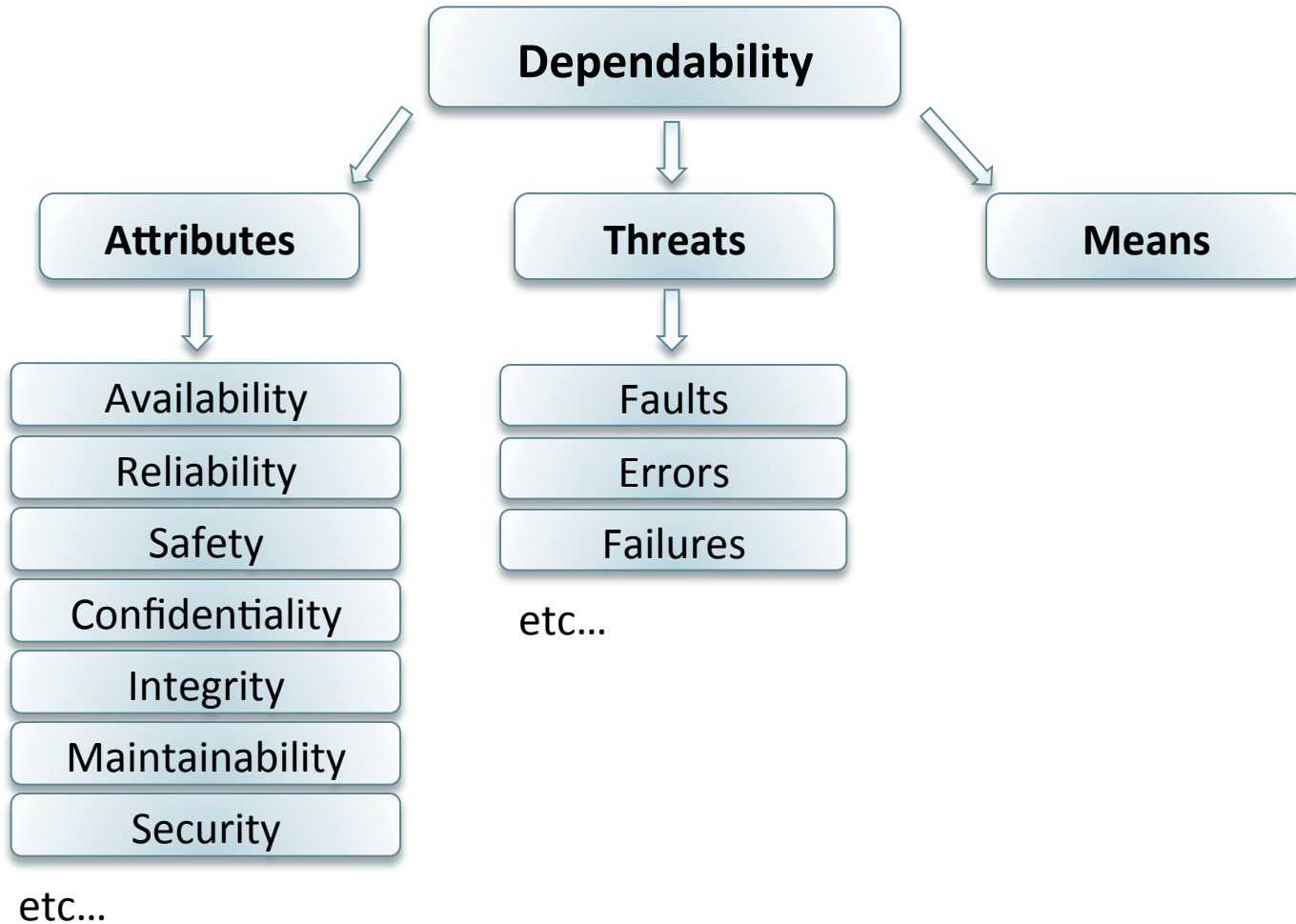
etc...

Dependability

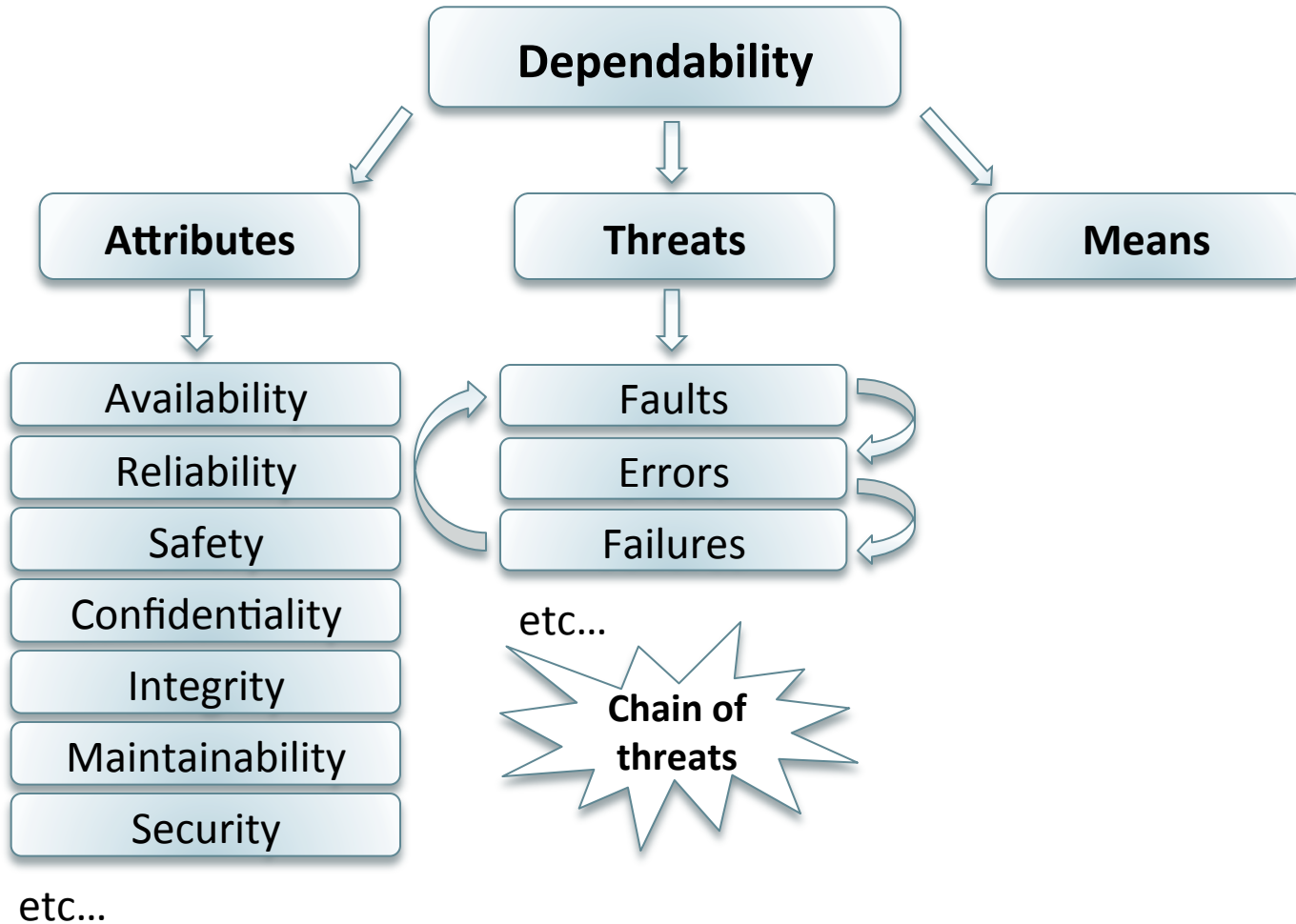


etc...

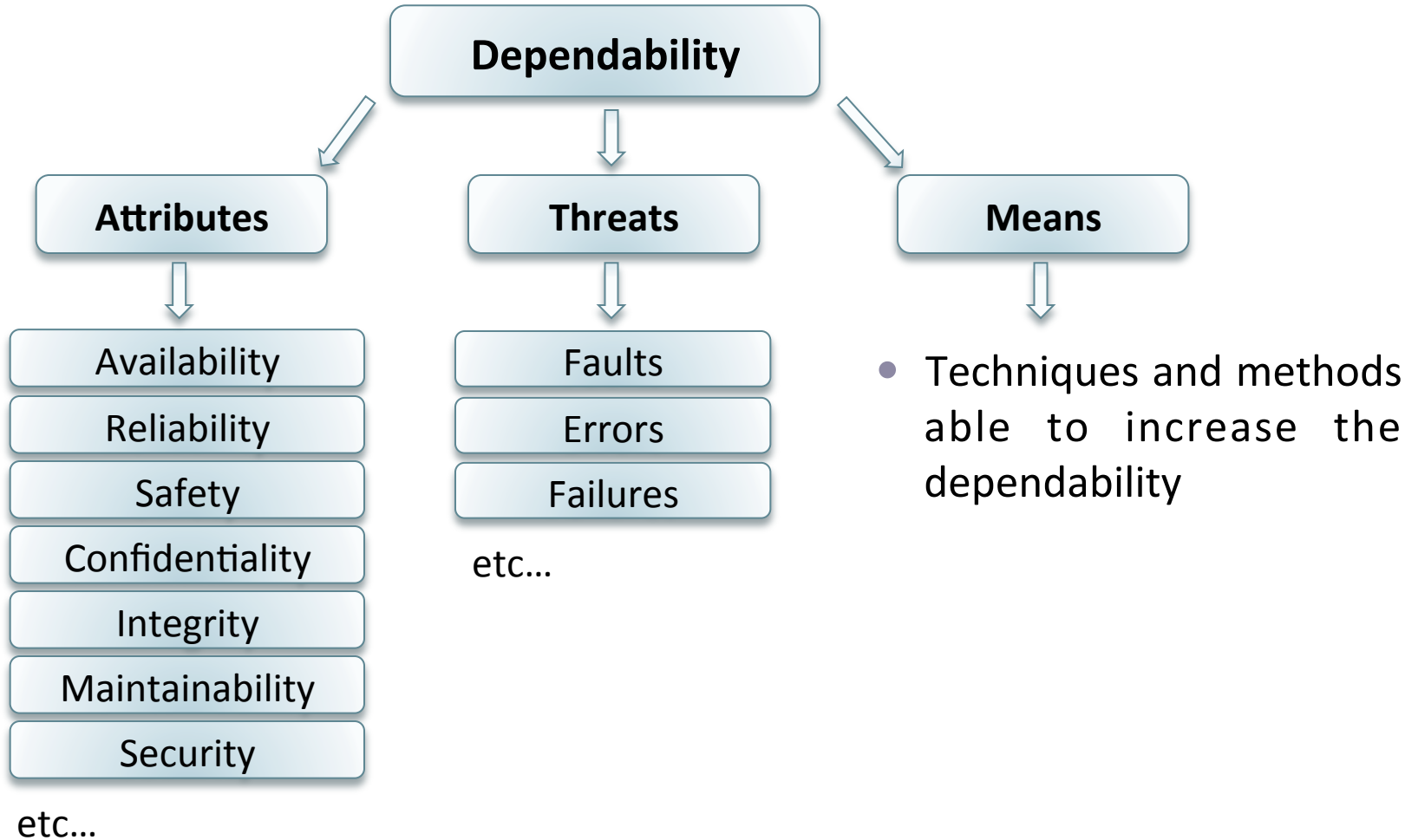
Dependability



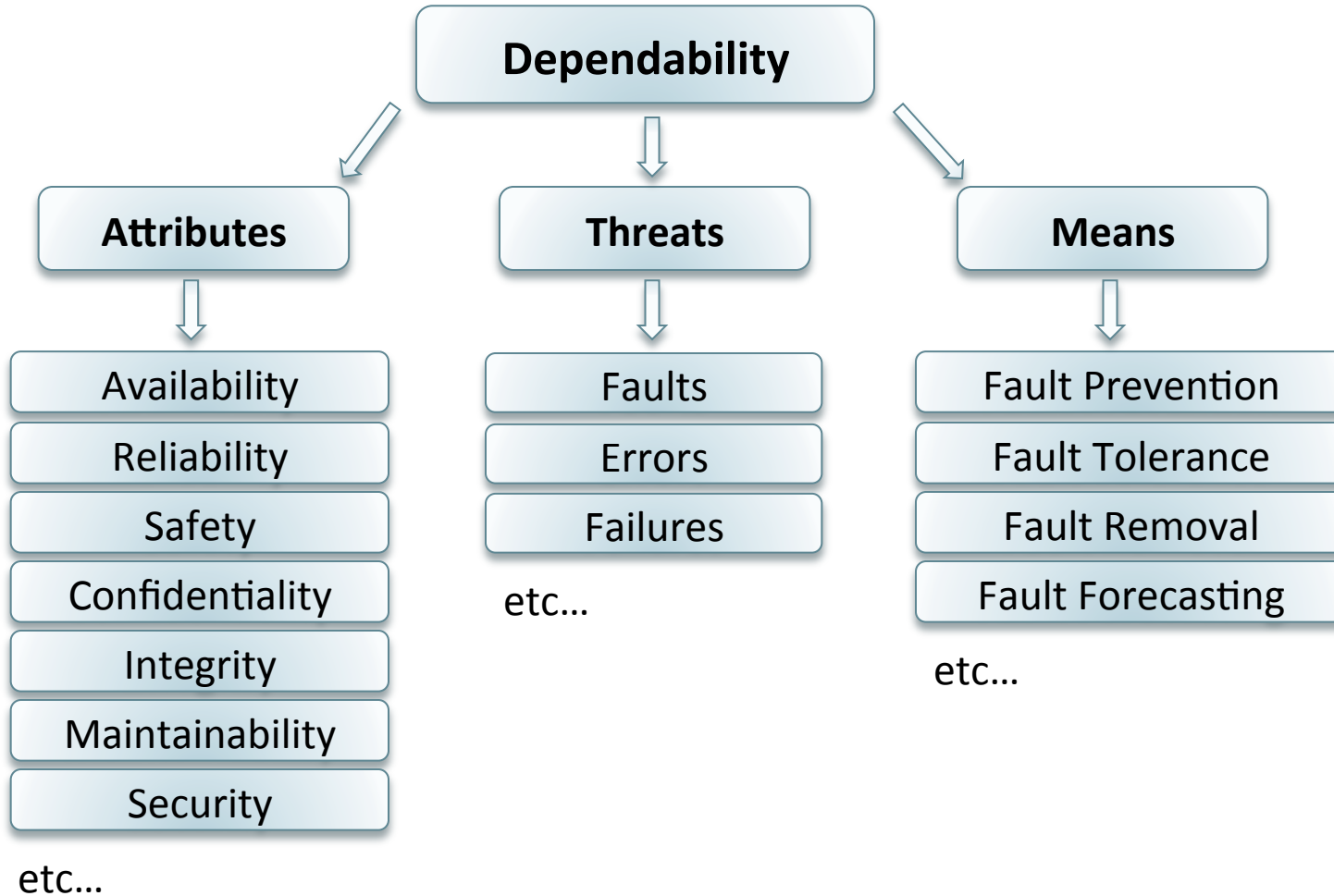
Dependability



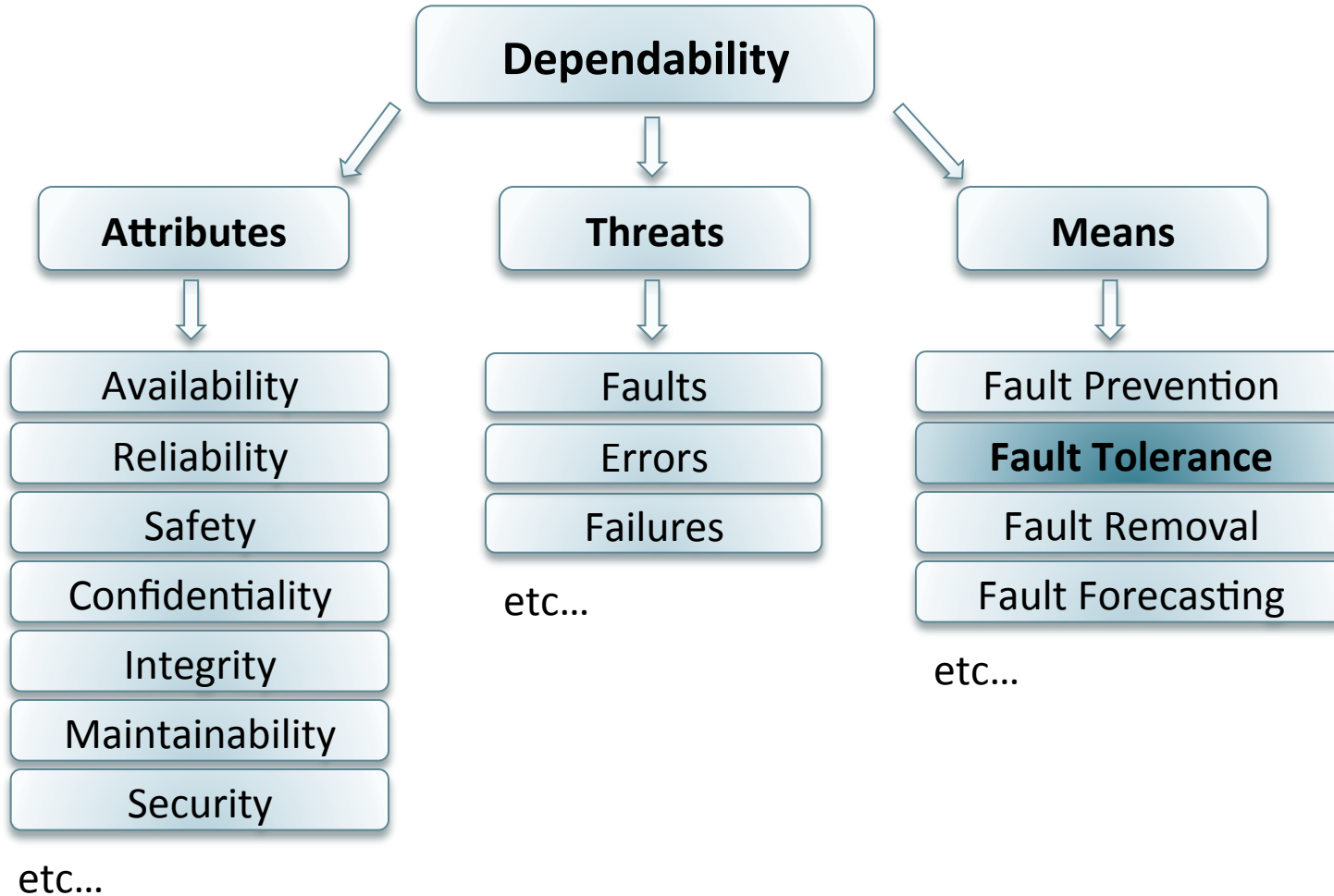
Dependability



Dependability



Dependability



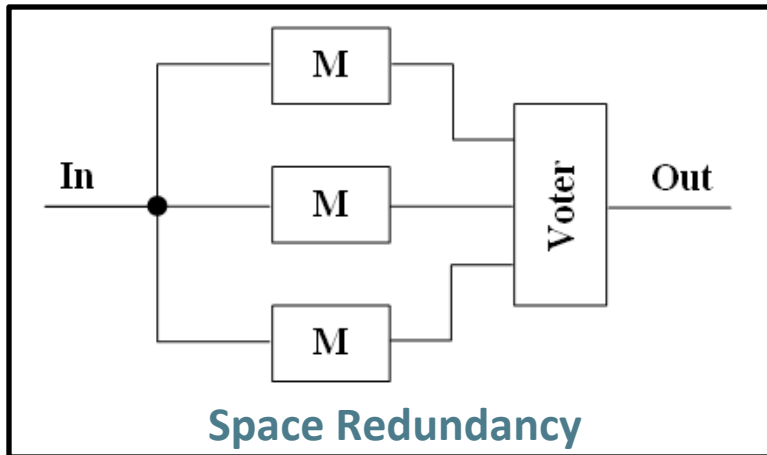
Outline

1. CLERECO Project
2. **State of the Art**
 - 2.1. Dependability
 - 2.2. **Fault Tolerance**
 - 2.3. Fault Injection
3. Research Direction
4. Conclusion and Perspective

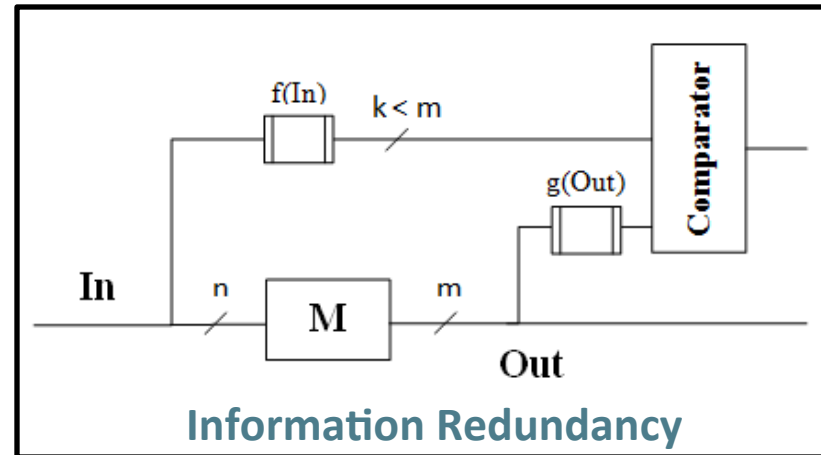
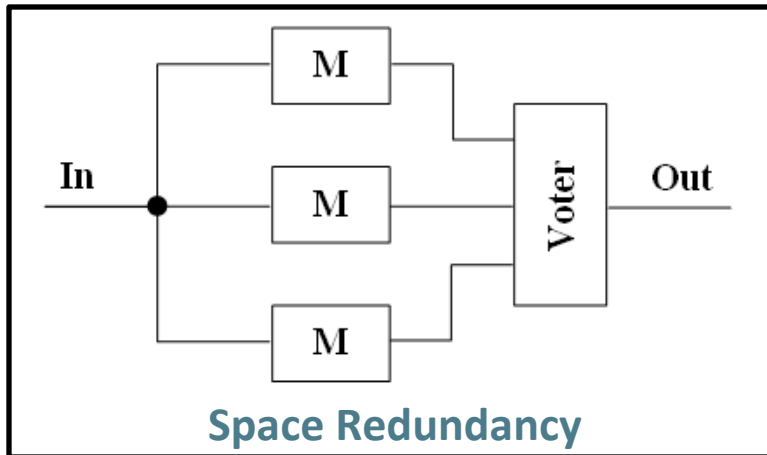
Fault Tolerance

- A method permitting to:
 - Increase the dependability
 - Avoid services failure in the presence of faults
- Several strategies used to tolerate faults, e.g. Redundancy.
- Redundancy is the use of additional hardware or software, not strictly necessary to functioning, but used in case of failure in other components.

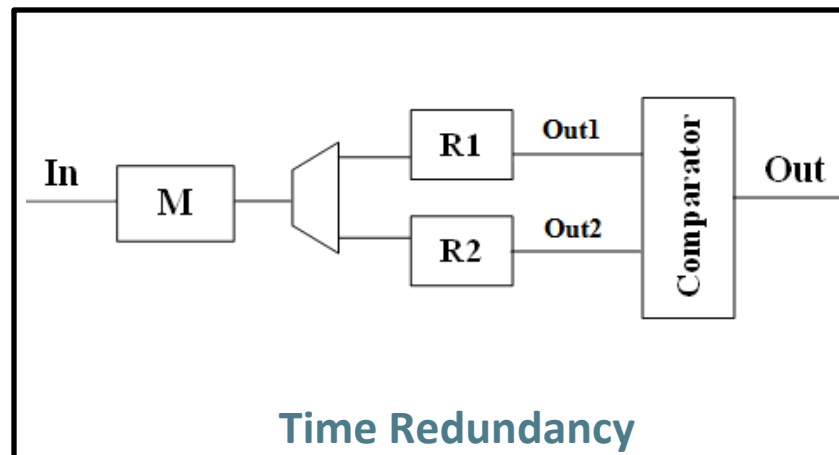
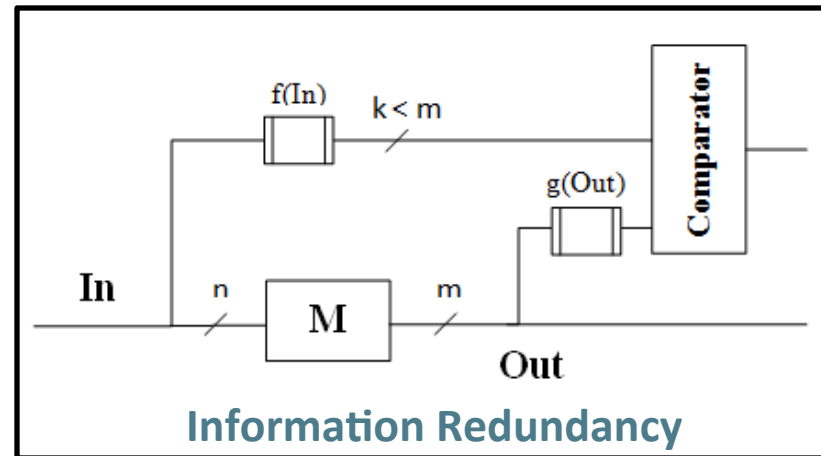
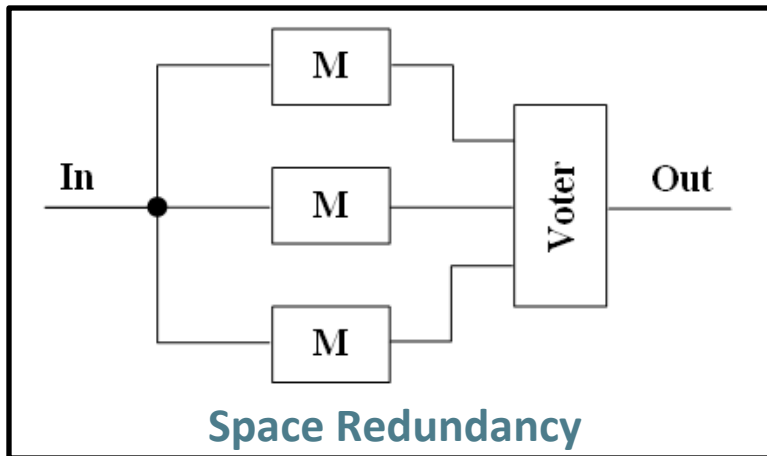
Redundancy





Redundancy





Redundancy





Fault Tolerance

| Techniques | Description | +/- | Examples |
|---|-------------|-----|----------|
| <p data-bbox="42 429 556 468">Hardware Fault Tolerance</p>  | | | |
| <p data-bbox="42 644 556 803">Software Implemented Hardware Fault Tolerance (SIHFT)</p>  | | | |



Fault Tolerance

| Techniques | Description | +/- | Examples |
|--|--|---|--|
| Hardware Fault Tolerance  | Changing the hardware to tolerate faults | <ul style="list-style-type: none">- Effective- Costly in term of equipment | <ul style="list-style-type: none">- Fault masking- Dynamic recovery |
| Software Implemented Hardware Fault Tolerance (SIHFT)  | | | |



Fault Tolerance

| Techniques | | Description | +/- | Examples |
|---|-------------------|--|--|---|
| Hardware Fault Tolerance  | | Changing the hardware to tolerate faults | <ul style="list-style-type: none"> - Effective - Costly in term of equipment | <ul style="list-style-type: none"> - Fault masking - Dynamic recovery |
| Software Implemented Hardware Fault Tolerance (SIHFT)  | OS Layer | | | |
| | Middleware Layer | | | |
| | Application Layer | | | |



Fault Tolerance

| Techniques | | Description | +/- | Examples |
|---|-------------------|--|--|---|
| Hardware Fault Tolerance  | | Changing the hardware to tolerate faults | <ul style="list-style-type: none"> - Effective - Costly in term of equipment | <ul style="list-style-type: none"> - Fault masking - Dynamic recovery |
| Software Implemented Hardware Fault Tolerance (SIHFT)  | OS Layer | Modifying the OS to achieve high level dependability | <ul style="list-style-type: none"> - Requires high skills to modify the OS | |
| | Middleware Layer | | | |
| | Application Layer | | | |

Fault Tolerance

| Techniques | | Description | +/- | Examples |
|---|-------------------|--|---|---|
| Hardware Fault Tolerance  | | Changing the hardware to tolerate faults | <ul style="list-style-type: none"> - Effective - Costly in term of equipment | <ul style="list-style-type: none"> - Fault masking - Dynamic recovery |
| Software Implemented Hardware Fault Tolerance (SIHFT)  | OS Layer | Modifying the OS to achieve high level dependability | <ul style="list-style-type: none"> - Requires high skills to modify the OS | |
| | Middleware Layer | Building an intermediate software layer to manage the communication between the OS and the application | <ul style="list-style-type: none"> - Efficient in case the application is not modifiable | <ul style="list-style-type: none"> - Interposition agents |
| | Application Layer | | | |

Fault Tolerance

| Techniques | | Description | +/- | Examples |
|---|-------------------|--|--|---|
| Hardware Fault Tolerance  | | Changing the hardware to tolerate faults | <ul style="list-style-type: none"> - Effective - Costly in term of equipment | <ul style="list-style-type: none"> - Fault masking - Dynamic recovery |
| Software Implemented Hardware Fault Tolerance (SIHFT)  | OS Layer | Modifying the OS to achieve high level dependability | <ul style="list-style-type: none"> - Requires high skills to modify the OS | |
| | Middleware Layer | Building an intermediate software layer to manage the communication between the OS and the application | <ul style="list-style-type: none"> - Efficient in case the application is not modifiable | <ul style="list-style-type: none"> - Interposition agents |
| | Application Layer | Acting directly on the software application | <ul style="list-style-type: none"> - Cheap - Better solution when the source code is available | <ul style="list-style-type: none"> - RECCO - CFCRE - ABFT |

Outline

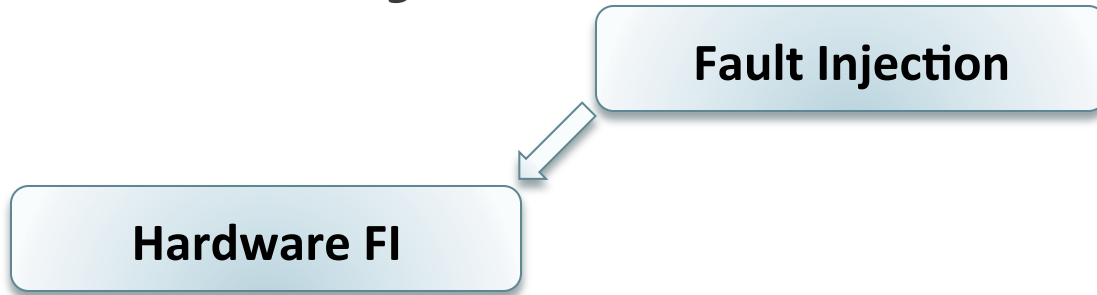
1. CLERECO Project
- 2. State of the Art**
 - 2.1. Dependability
 - 2.2. Fault Tolerance
 - 2.3. Fault Injection**
3. Research Direction
4. Conclusion and Perspective

Fault Injection

Fault Injection

- A validation technique of the dependability for fault tolerance systems
- Evaluate the behavior of the system in the presence of faults

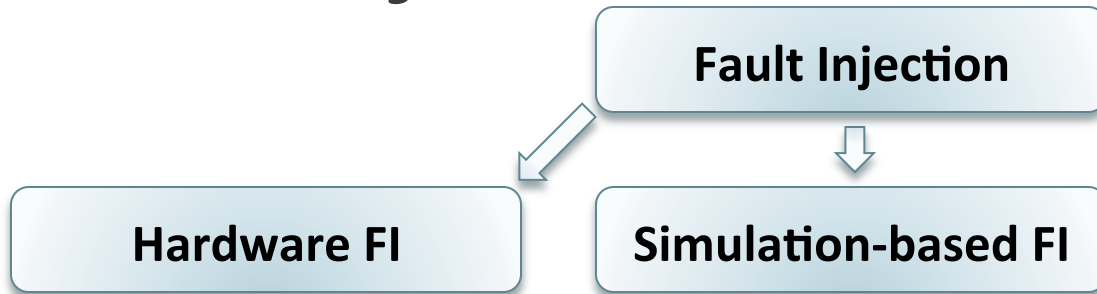
Fault Injection



+ High time-resolution.

- Expensive in term of equipment.
- Risk to damage the system.

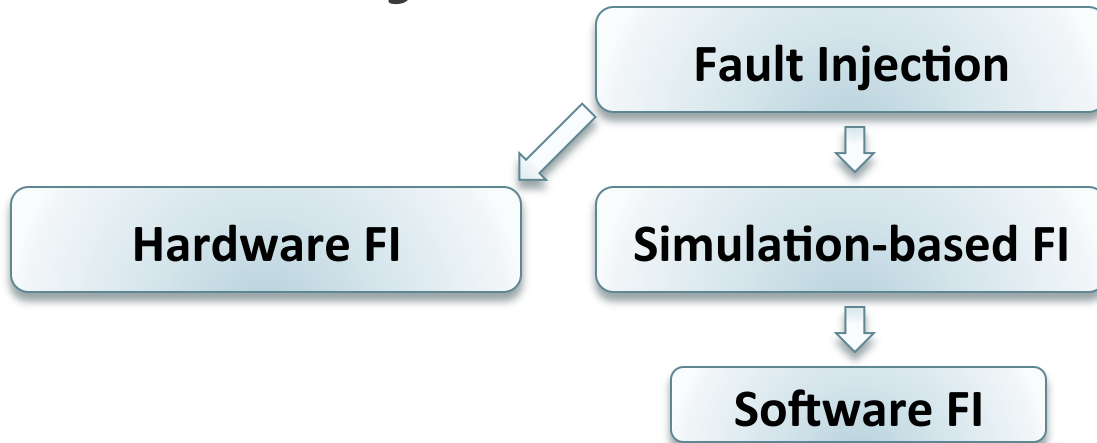
Fault Injection



- + High time-resolution.
- Expensive in term of equipment.
- Risk to damage the system.

- + No risk to damage the system
- + Low-cost
- + Simple to set-up
- Accuracy of fault model and system model

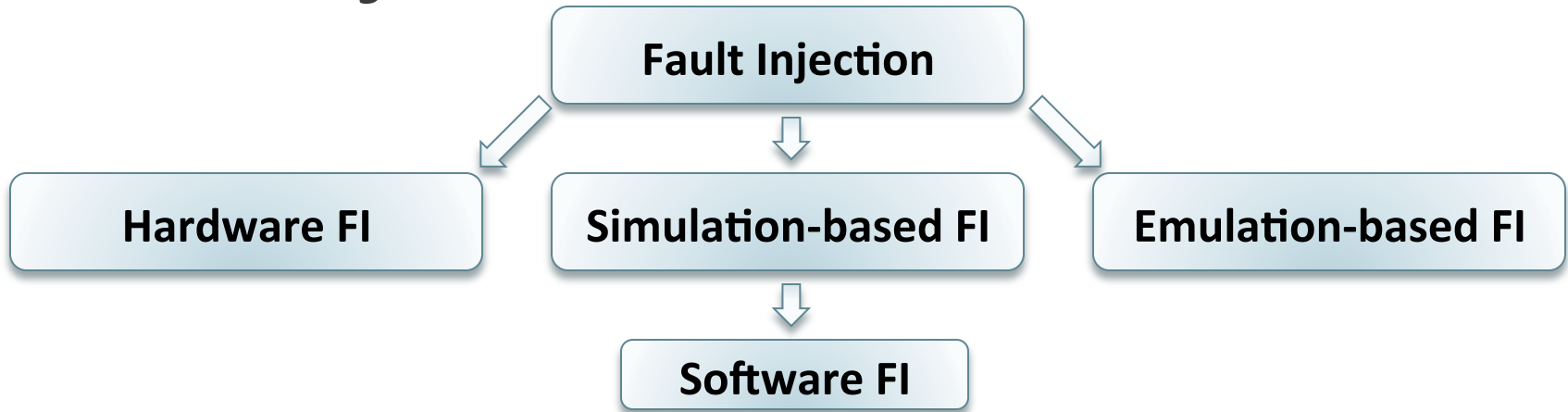
Fault Injection



- + High time-resolution.
- Expensive in term of equipment.
- Risk to damage the system.

- + No risk to damage the system
- + Low-cost
- + Simple to set-up
- Accuracy of fault model and system model

Fault Injection



- + High time-resolution.
- Expensive in term of equipment.
- Risk to damage the system.

- + No risk to damage the system
- + Low-cost
- + Simple to set-up
- Accuracy of fault model and system model

- + It reduces the execution time compared to the simulation-based FI.
- Costly and not flexible

Fault Injection Environments

| Tool | Developer | Category | Description |
|-------------|--------------------------------|-------------|--|
| Jaca | University of Campinas, Brazil | Software FI | <ul style="list-style-type: none">• Faults are injected in object-oriented systems.• Adapted to any Java application. |

Fault Injection Environments

| Tool | Developer | Category | Description |
|-----------------|---------------------------------|-------------|---|
| Jaca | University of Campinas, Brazil | Software FI | <ul style="list-style-type: none">• Faults are injected in object-oriented systems.• Adapted to any Java application. |
| Xception | University of Coimbra, Portugal | Software FI | <ul style="list-style-type: none">• Fault are injected in software• It uses advanced debugging features to observe the behavior of the system in detail in presence of faults. |

Fault Injection Environments

| Tool | Developer | Category | Description |
|-----------------|---------------------------------|-------------|---|
| Jaca | University of Campinas, Brazil | Software FI | <ul style="list-style-type: none">• Faults are injected in object-oriented systems.• Adapted to any Java application. |
| Xception | University of Coimbra, Portugal | Software FI | <ul style="list-style-type: none">• Fault are injected in software• It uses advanced debugging features to observe the behavior of the system in detail in presence of faults. |
| RIFLE | University of Coimbra, Portugal | Hardware FI | <ul style="list-style-type: none">• Faults are injected in pin-level of the modules.• It performs analysis to observe the impact of faults on the processor |

Fault Injection Environments

| Tool | Developer | Category | Description |
|-----------------|---------------------------------|---------------------|---|
| Jaca | University of Campinas, Brazil | Software FI | <ul style="list-style-type: none">• Faults are injected in object-oriented systems.• Adapted to any Java application. |
| Xception | University of Coimbra, Portugal | Software FI | <ul style="list-style-type: none">• Fault are injected in software• It uses advanced debugging features to observe the behavior of the system in detail in presence of faults. |
| RIFLE | University of Coimbra, Portugal | Hardware FI | <ul style="list-style-type: none">• Faults are injected in pin-level of the modules.• It performs analysis to observe the impact of faults on the processor |
| LIFTING | LIRMM Montpellier, France | Simulation-based FI | <ul style="list-style-type: none">• A simulator able to perform both logic and fault simulation for stuck-at faults and single event-upset on digital circuits• It provides many features to analyze the fault simulation results.• It allows describing the hardware systems to define the software stored in the memory, and to inject faults in the hardware model elements. |

Outline

1. CLERECO Project
2. State of the Art
 - 2.1. Dependability
 - 2.2. Fault Tolerance
 - 2.3. Fault Injection
- 3. Research Direction**
4. Conclusion and Perspective

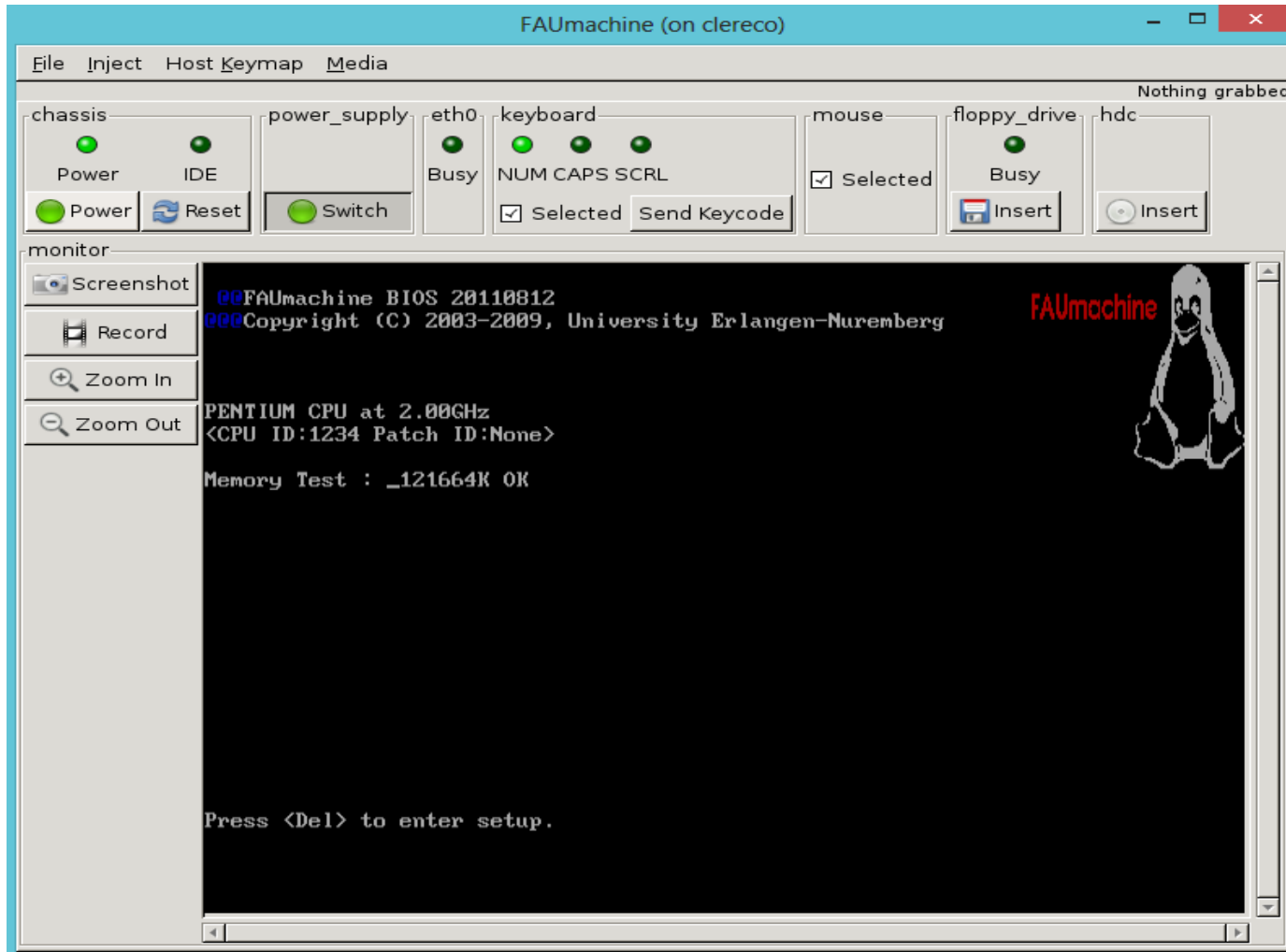
FAUMachine open source



University of Nuremberg,
Germany

- A virtual machine similar to QEMU or Virtual Box.
- It permits to inject faults and observe the whole operating system or application software.
- It performs with a high simulation speed thanks to the virtualization.

FAUMachine Installation



Virtual Machine

- The FAUmachine virtual machine runs as a normal user process on top of Linux on i386 and AMD64 hardware.
- $\frac{1}{4}$ of the performance of the host system:
 - CPU/Memory/ROM: 5 times slower
 - Disk: 3 times slower
 - Network: 2 times slower

Which Type of Faults?

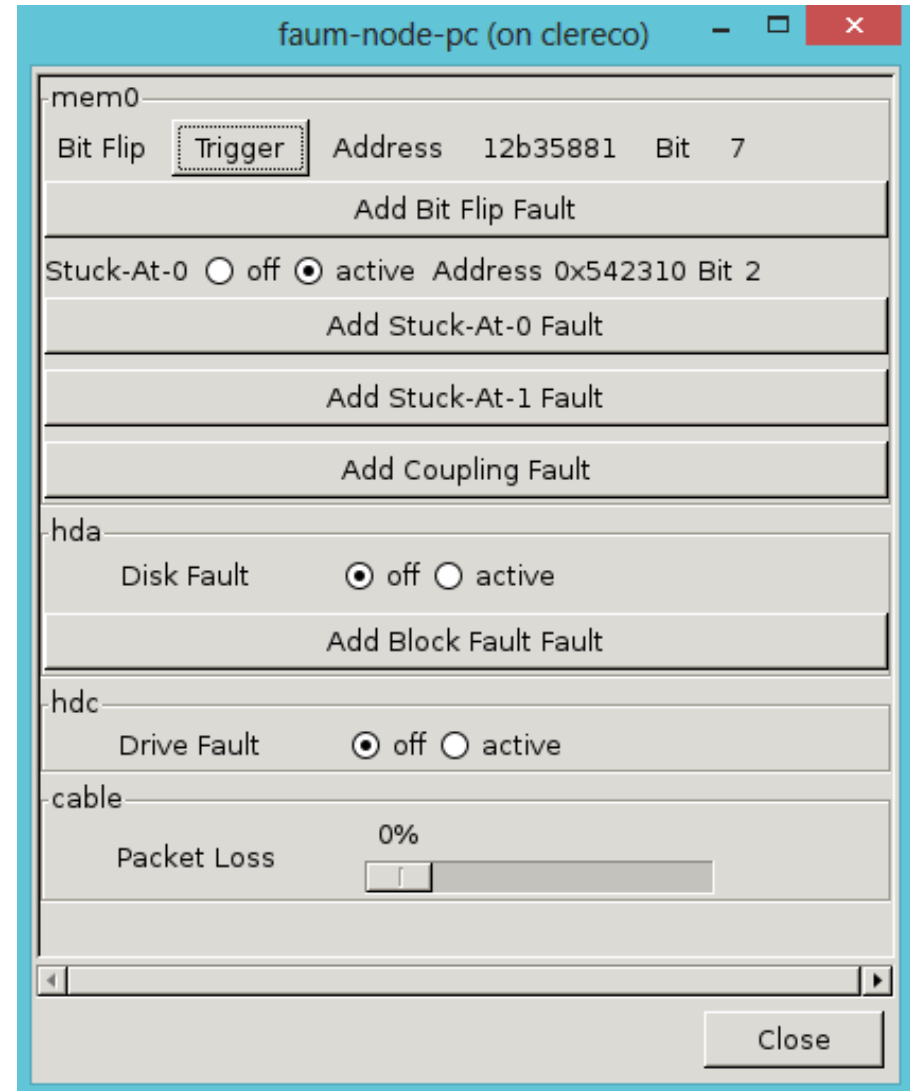
- **Memory Cells/CPU Register:**
 - Transient bit flips
 - Permanent stuck-at faults
 - Permanent coupling faults
- **Disk/CDROM:**
 - Transient / permanent block faults
 - Transient / permanent whole disk faults
- **Network:**
 - Transient send / receive faults
 - Intermittent send / receive faults
 - Permanent send / receive faults

How to Inject Faults?

- Via GUI in FAUMachine

How to Inject Faults?

- Via GUI in FAUMachine



How to Inject Faults?

- Via GUI in FAUMachine
- Via VHDL script
 - Define faults type, location, time and duration

How to Inject Faults?

- Via GUI in FAUMachine
- Via VHDL script
 - Define faults type, location, time and duration

```
architecture behaviour of fi is
    signal err : boolean;
begin
    process
    begin
        shortcut_bool_out(
            err,
            ":pc:mem0",
            "stuck_at_0/0x543210/0");
        err <= true;
    end process;
end behaviour;
```

How to Inject Faults?

- Via GUI in FAUMachine
- Via VHDL script
 - Define faults type, location, time and duration

Define the signal for
the fault

```
architecture behaviour of fi is
    → signal err : boolean;
begin
    process
    begin
        shortcut_bool_out(
            err,
            ":pc:mem0",
            "stuck_at_0/0x543210/0");
        err <= true;
    end process;
end behaviour;
```

How to Inject Faults?

- Via GUI in FAUMachine
- Via VHDL script
 - Define faults type, location, time and duration

```
architecture behaviour of fi is
    signal err : boolean;
begin
    process
    begin
        shortcut_bool_out(
            err,
            ":pc:mem0",
            "stuck_at_0/0x543210/0");
        err <= true;
    end process;
end behaviour;
```

Connect the actual fault



How to Inject Faults?

- Via GUI in FAUMachine
- Via VHDL script
 - Define faults type, location, time and duration

The signal of the fault

```
architecture behaviour of fi is
    signal err : boolean;
begin
    process
    begin
        shortcut_bool_out(
            > err,
            ":pc:mem0",
            "stuck_at_0/0x543210/0");
        err <= true;
    end process;
end behaviour;
```

How to Inject Faults?

- Via GUI in FAUMachine
- Via VHDL script
 - Define faults type, location, time and duration

```
architecture behaviour of fi is
    signal err : boolean;
begin
    process
    begin
        shortcut_bool_out(
            err,
            → ":pc:mem0",
            "stuck_at_0/0x543210/0");
        err <= true;
    end process;
end behaviour;
```

The path to the
instantiated
component

How to Inject Faults?

- Via GUI in FAUMachine
- Via VHDL script
 - Define faults type, location, time and duration

Fault Parameters

```
architecture behaviour of fi is
    signal err : boolean;
begin
    process
    begin
        shortcut_bool_out(
            err,
            ":pc:mem0",
            → "stuck_at_0/0x543210/0");
        err <= true;
    end process;
end behaviour;
```

How to Inject Faults?

- Via GUI in FAUMachine
- Via VHDL script
 - Define faults type, location, time and duration

```
architecture behaviour of fi is
    signal err : boolean;
begin
    process
    begin
        shortcut_bool_out(
            err,
            ":pc:mem0",
            "stuck_at_0/0x543210/0");
        err <= true;
    end process;
end behaviour;
```

Activate the fault



Outline

1. CLERECO Project
2. State of the Art
 - 2.1. Dependability
 - 2.2. Fault Tolerance
 - 2.3. Fault Injection
3. Research Direction
4. **Conclusion and Perspective**

Conclusion and Perspective

- FAUMachine:
 - Inject faults using VHDL script
 - Observe the impact of faults
- LLVM and its based fault injection tool LLFI



Thank you