Fly me to the moon …

Professor Mike Hinchey

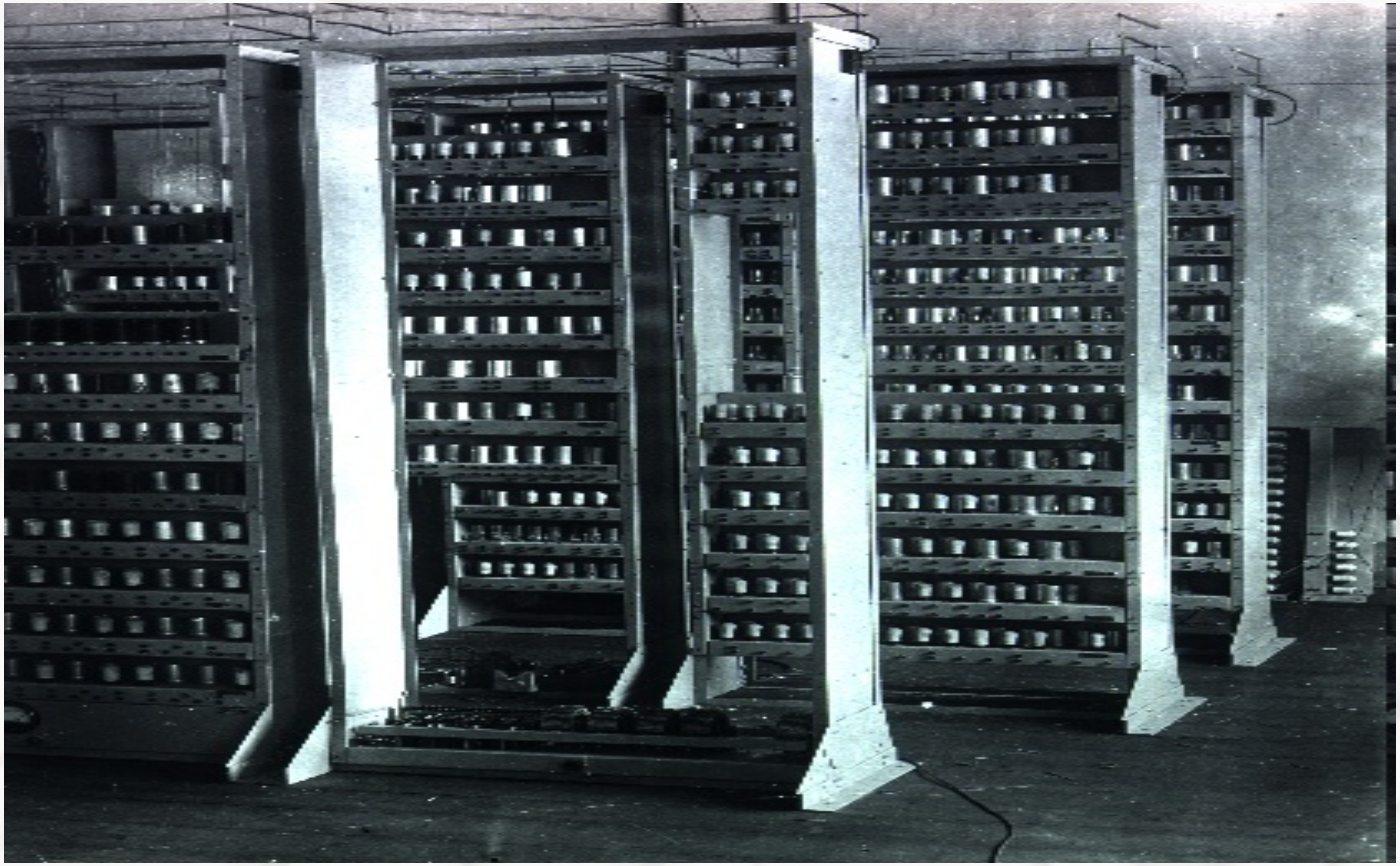
sfi science foundation ireland
fondúireacht eolaíochta éireann

HEA
Higher Education Authority
An tÚdarás um Ard-Oideachas

NDP
PLEAN FORBARTHA NÁISIÚNTA


MAD
MAD
MAD
MAD
MAD
MAD

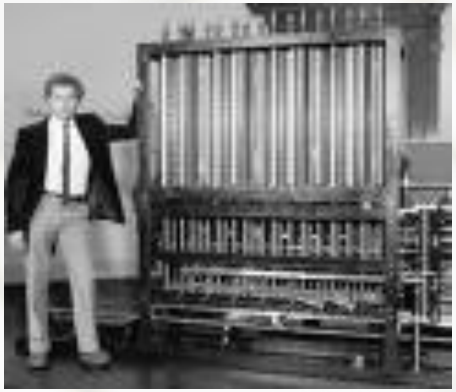
66 Years Ago …



EDSAC

- 650 instructions per second.
- 1024 17-bit words of memory in mercury ultrasonic delay lines.
- Paper tape input and teleprinter output at 6 2/3 characters per second.
- 3000 valves, 12 kW power consumption, occupied a room 5m by 4m.
- "Operating system" occupied 31 words of read-only memory.
- Early use to solve problems in meteorology, genetics and X-ray crystallography.

Difference Engine



Motivation

Errata, detected in Taylor's Logarithms. *London: 4to, 1972 [sic]*

…

6 Kk Co-sine of 14.18.3 – 3398 – 3298

Nautical Almanac (1832)

…

In the list of ERRATA detected in Taylor's *Logarithms*, for cos. 4 18' 3'' read cos. 14 18'2''.

Nautical Almanac (1833)

ERRATUM of the ERRATUM of the ERRATA of TAYLOR'S *Logarithms.* For cos. 4 18'3'', *read* 14 18' 3''.

Nautical Almanac (1836)

First Programmer

Augusta Ada King, Countess of Lovelace

Challenges for Software Engineering

- Increases in demand for greater, more complex functionality;
- Stricter (required and desirable) constraints on performance and reaction times;
- Attempts to increase productivity and reduce costs while constantly pushing requirements to the limit;
- Requirement of regular change and evolving systems.

Evolution

Any intelligent fool can make things bigger and more complex …

It takes a touch of genius and a lot of courage to move in the opposite direction.

Albert Einstein

Evolving systems are software systems which exhibit change over time.

Software is supposed to change…
otherwise it would be in the hardware!

Evolving Critical Systems

- have evolved from legacy code and legacy systems, or
- result from a combination of existing component-based systems, possibly over significant periods of time, or
- evolve as a result of a focused and intentional change in organization and architecture to exploit newer techniques believed to be beneficial;
- they require that the system adapt and evolve at run-time in order to react to changes in the environment or to meet necessary constraints on the system that were not previously satisfied and possibly not previously known.

Critical Systems

Critical systems are systems where

- failure or malfunction will lead to significant negative consequences;
- these systems may have strict requirements for security and safety, to protect the user or others;
- alternatively, these systems may be critical to the organization's mission, product base, profitability or competitive advantage.

ECS Research Agenda

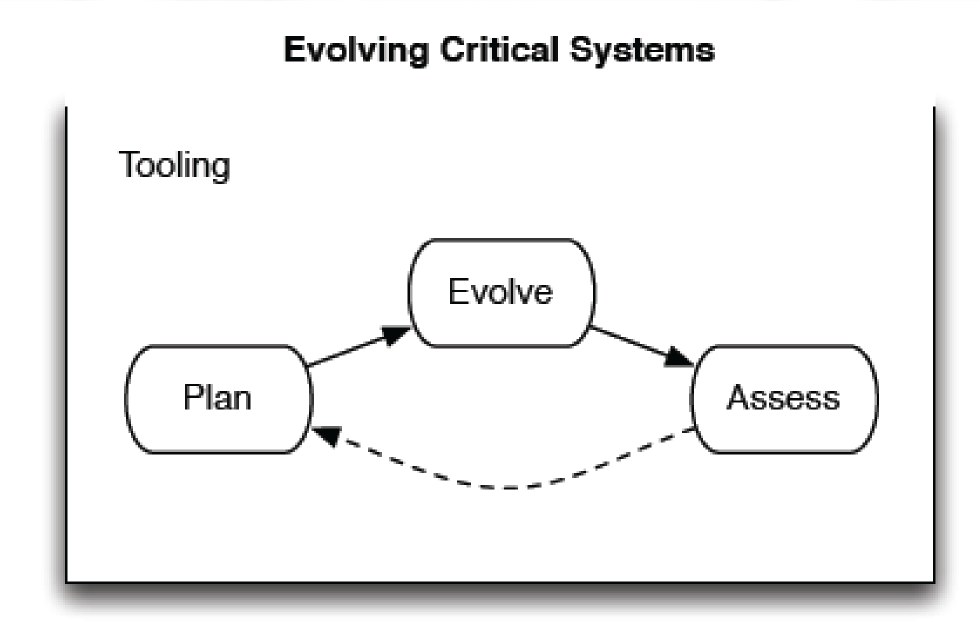
An ECS Research Agenda addresses several core research topics in the evolving critical systems field.

The central research topic is **building software that**

(a) is highly reliable, and

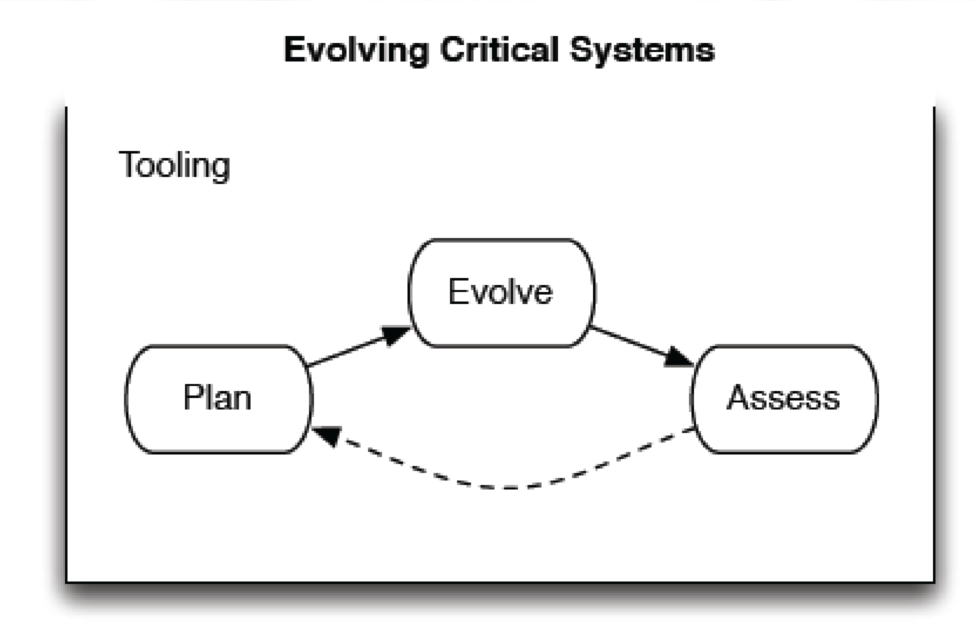
(b) retains this reliability as it evolves, *without* incurring prohibitive costs.

PEA+T

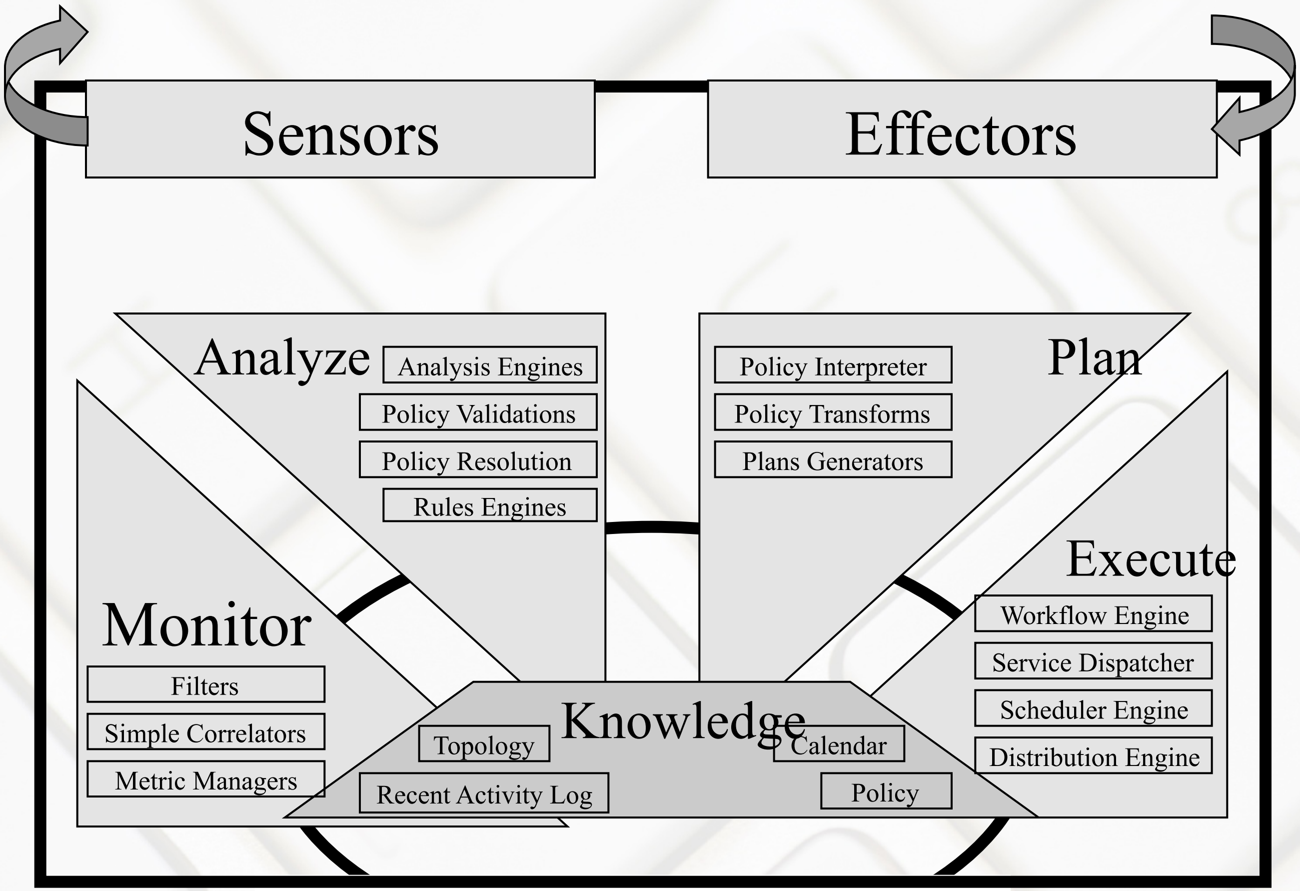


Peat

PEA+T



MAPE



Source: IBM, AC Blueprint 2003

Evolving Critical Systems

Evolving

- Software is meant to change, both at design/revision time and at run-time
- Lero's research focuses on methods and tools for designing software that can be changed or that can change itself without degradation

Critical

- Much of today's software is mission-, safety- or business-critical
- Lero is researching methods and tools to improve the integrity of and confidence levels in critical software

Systems

- Expertise in software engineering needs to be coupled with domain knowledge
- Lero has existing partnerships & expertise in medical devices, space, future cities and financial services
- We will continue to seek domain-related partners for collaborative r

Lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

Key Focus Areas

- A: Methods & Standards for High Integrity Systems
  - Lean, Agile & Global methods
  - Open Sourcing & Innovation
  - Standardised SW Development processes
  - Model-based approaches
  - Formal methods & safety use cases
- B: Adaptive & Autonomous Systems
  - Systems that learn & respond to their environments
- C: Software Performance
  - Large complex systems
  - Multicore embedded & massively parallel systems
- D: Security & Privacy
  - New approaches to security and privacy and the trade-offs between them
  - Digital forensics

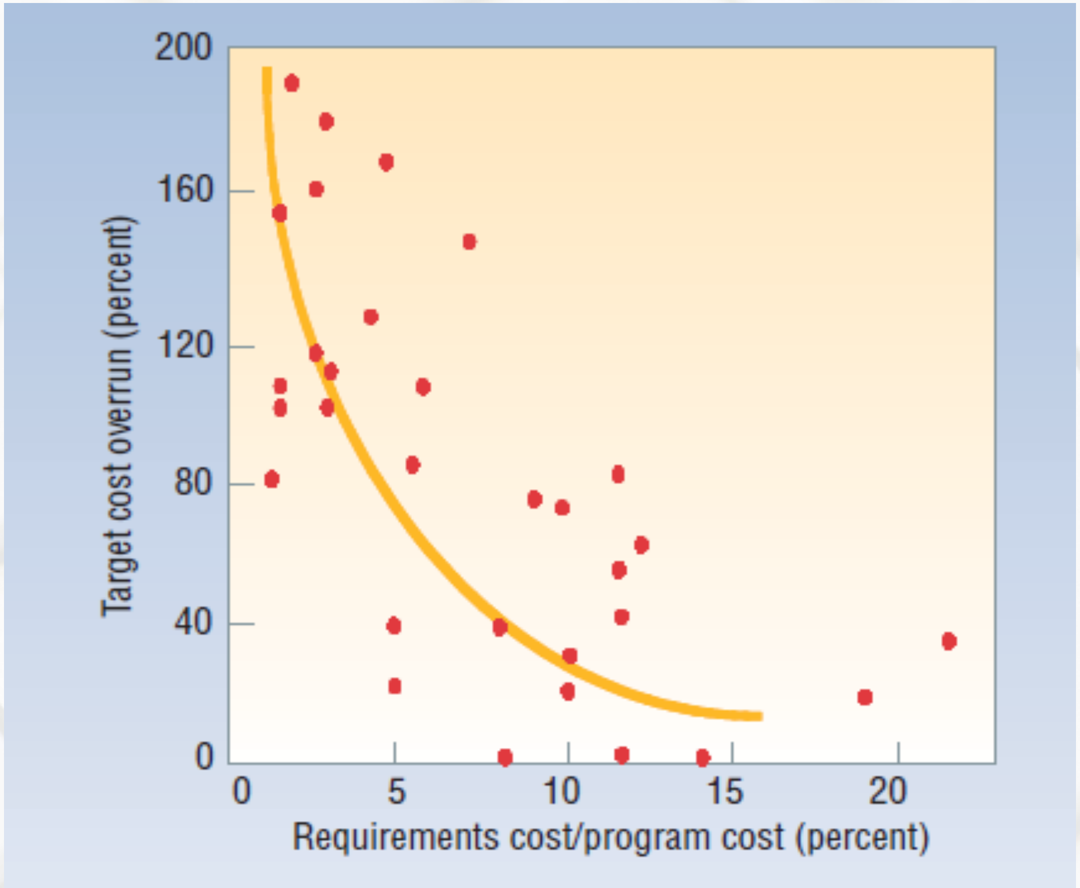
lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

Hub and Spoke Model

An ECS Scenario

- Space Exploration
  - Some of the most complex and expensive software applications to date.
  - High Levels of Autonomy.
  - Significant consequences for failure.

Requirements Effort vs. Cost Overrun



Swarm Technologies

- Inspired by swarms of bees and flocks of birds in nature;
- Many application areas:
  - drug discovery;
  - communication systems;
  - environmental monitoring;
  - exploration.

Coordinated swarms of smaller spacecraft will offer:

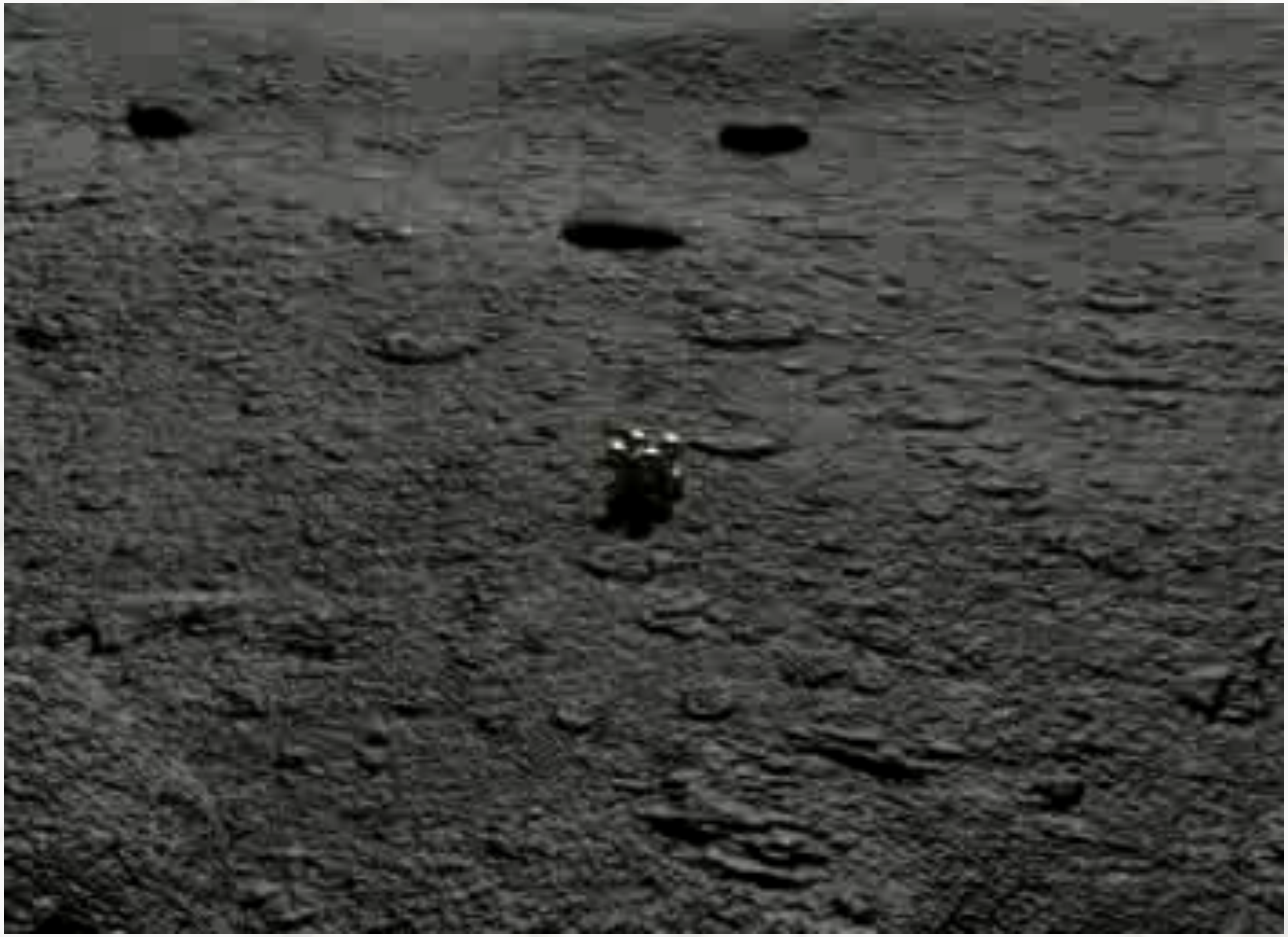
- More effective use of solar power;
- Access to areas where large craft could not go;
- Ability to perform more complex tasks;
- Greater accuracy and flexibility.

Autonomous NanoTechnology Swarm

Three concept sub-missions:

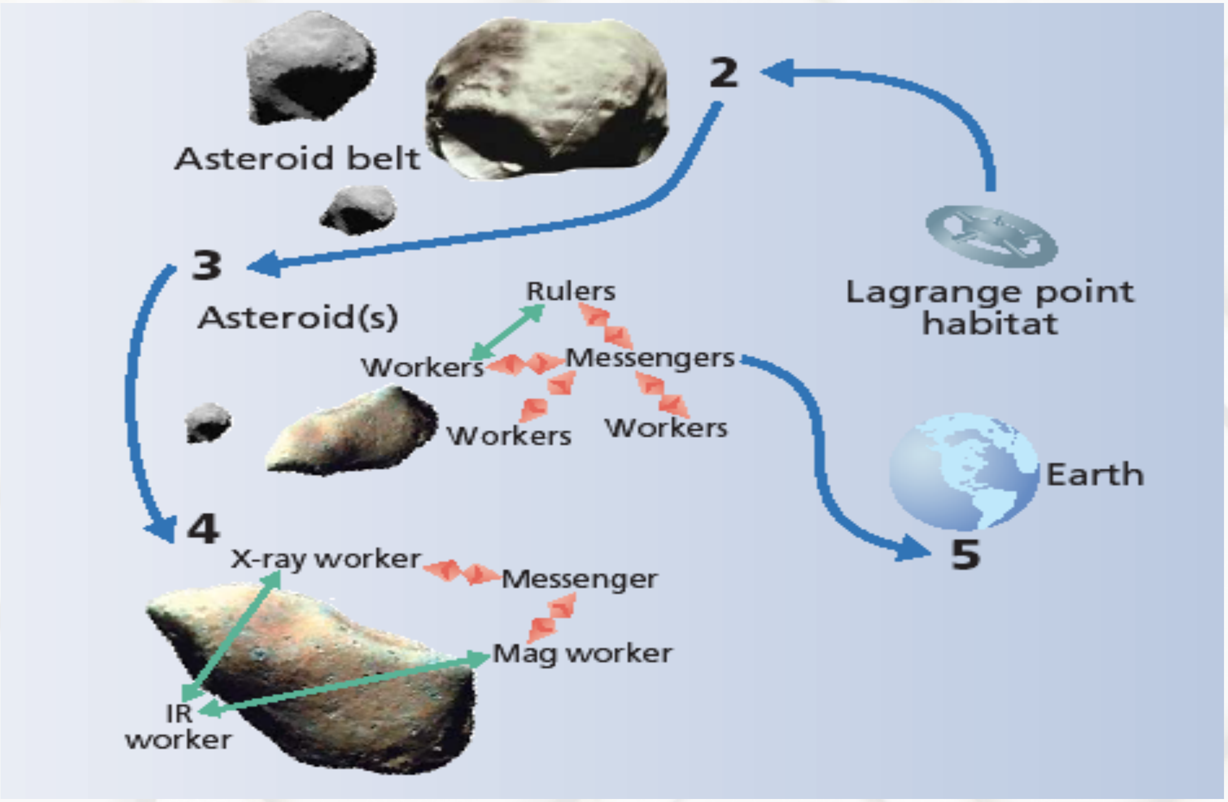
1. Lander Amorphous Rover Antenna (LARA)
2. Saturn Autonomous Ring Array (SARA)
3. Prospecting Asteroid Mission (PAM)

Tet Walkers

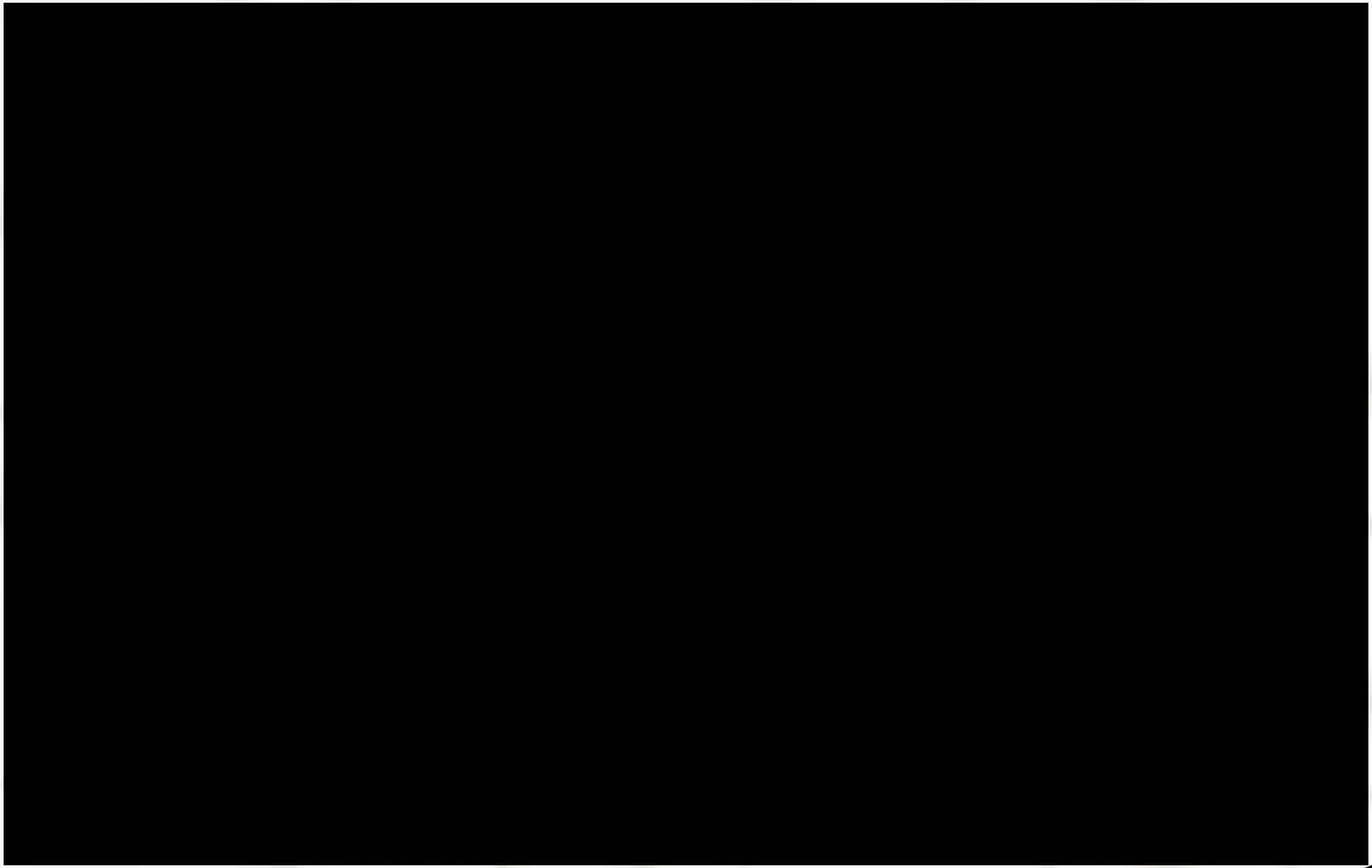


LARA Walkers

ANTS Concept Mission - PAM



PAM : the Movie



ECS Contributions

1. Formal Methods
2. Autonomic Computing
3. Software Product Lines
4. Automatic Code Generation

Model of Formal Method

v

Swarm Formal Method Model and Outline

$$LEADER\_COM_{i,conv} = leader.in?msg \rightarrow \text{case } \begin{cases} LEADER\_MESSAGE_{i,conv,msg} & \text{if sender}(msg) = LEADER \\ MESSENGER\_MESSAGE_{i,conv,msg} & \text{if sender}(msg) = MESSENGER \\ WORKER\_MESSAGE_{i,conv,msg} & \text{if sender}(msg) = WORKER \\ EARTH\_MESSAGE_{i,conv,msg} & \text{if sender}(msg) = EARTH \\ ERROR\_MESSAGE_{i,conv,msg} & otherwise \end{cases}$$

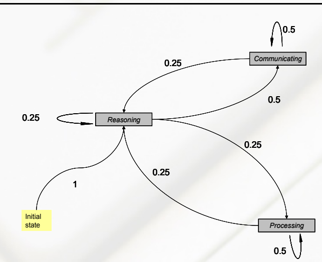
$$\Phi = \begin{Bmatrix} SendMessage, ReceiveMessage, \\ Reason, Process \end{Bmatrix}$$

is a set of (partial) transition functions where each transition function maps

$$Memory \times Input \rightarrow Output \times Memory$$

$$memory' = (Goals', Model', CommsTracking)$$

[Communicating]ReasoningDeliberatve(Leader)[Reasoning]
[Reasoning] SendMessage (Leader,Worker)[Communicating]
[Processing] SendMessage (Leader, Worker) [Communicating]



$$P = \begin{pmatrix} 0 & .5 & .25 & .25 \\ 0 & .5 & .25 & .25 \\ 0 & .5 & .25 & .25 \\ 0 & 0 & .5 & .5 \end{pmatrix}$$

Agent State	Actions leading to the agent state	f	p
	Identity		
Communicating	SendMessageWorker	50	2
	SendMessageLeader	50	2
	SendMessageError	1	1
	ReceiveMessageWorker	50	2
	ReceiveMessageLeader	50	2
	ReceiveMessageError	1	1
Reasoning	ReasoningDeliberatve	50	2
	ReasoningReactive	50	2
Processing	ProcessingSortingAndStorage	17	2
	ProcessingGeneration	17	2
	ProcessingPrediction	17	2
	ProcessingDiagnosis	16	2
	ProcessingRecovery	16	2
	ProcessingRemediation	17	2

$$\begin{aligned} Communicating = {} & 50\omega^2 : \text{SendMessageWorker}.Communicating \\ & + 50\omega^2 : \text{SendMessageLeader}.Communicating \\ & + 1\omega^1 : \text{SendMessageError}.Communicating \\ & + 50\omega^2 : \text{ReceiveMessageWorker}.Communicating \\ & + 50\omega^2 : \text{ReceiveMessageLeader}.Communicating \\ & + 1\omega^1 : \text{ReceiveMessageError}.Communicating \\ & + 50\omega^2 : \text{ReasoningDeliberatve}.Reasoning \\ & + 50\omega^2 : \text{ReasoningReactive}.Reasoning \\ & + 17\omega^2 : \text{ProcessingSortingAndStorage}.Processing \\ & + 17\omega^2 : \text{ProcessingGeneration}.Processing \\ & + 17\omega^2 : \text{ProcessingPrediction}.Processing \\ & + 16\omega^2 : \text{ProcessingDiagnosis}.Processing \\ & + 16\omega^2 : \text{ProcessingRecovery}.Processing \\ & + 17\omega^2 : \text{ProcessingRemediation}.Processing \end{aligned}$$

$$n\omega^{k+l} + m\omega^k = n\omega^{k+l} = m\omega^k + n\omega^{k+l}$$

$$n\omega^k + m\omega^k = (n+m)\omega^k = m\omega^k + n\omega^k$$

ASSL Specification

```
AEIP {
   MESSAGES { ... }
   CHANNELS { ... }
   FUNCTIONS { ... }
   MANAGED_ELEMENTS {
      MANAGED_ELEMENT worker {
         INTERFACE_FUNCTION getDistanceToNearestObject { RETURNS { DECIMAL } }
      }
   }
} // AEIP

METRICS {
   METRIC distanceToNearestObject  {
      METRIC_TYPE  {  RESOURCE  }
      METRIC_SOURCE { AEIP.MANAGED_ELEMENTS.worker.getDistanceToNearestObject }
      DESCRIPTION  { "measures the distance to the nearest space object" }
      MEASURE_UNIT  { "KM" }
      VALUE { 100 }
      THRESHOLD_CLASS  {  DECIMAL  [0.001 ~ )  }
   }
}
```

ECS Contributions

1. Formal Methods
2. Autonomic Computing
3. Software Product Lines
4. Automatic Code Generation

Autonomic Computing

Inspiration from the human/mammalian autonomic nervous system.

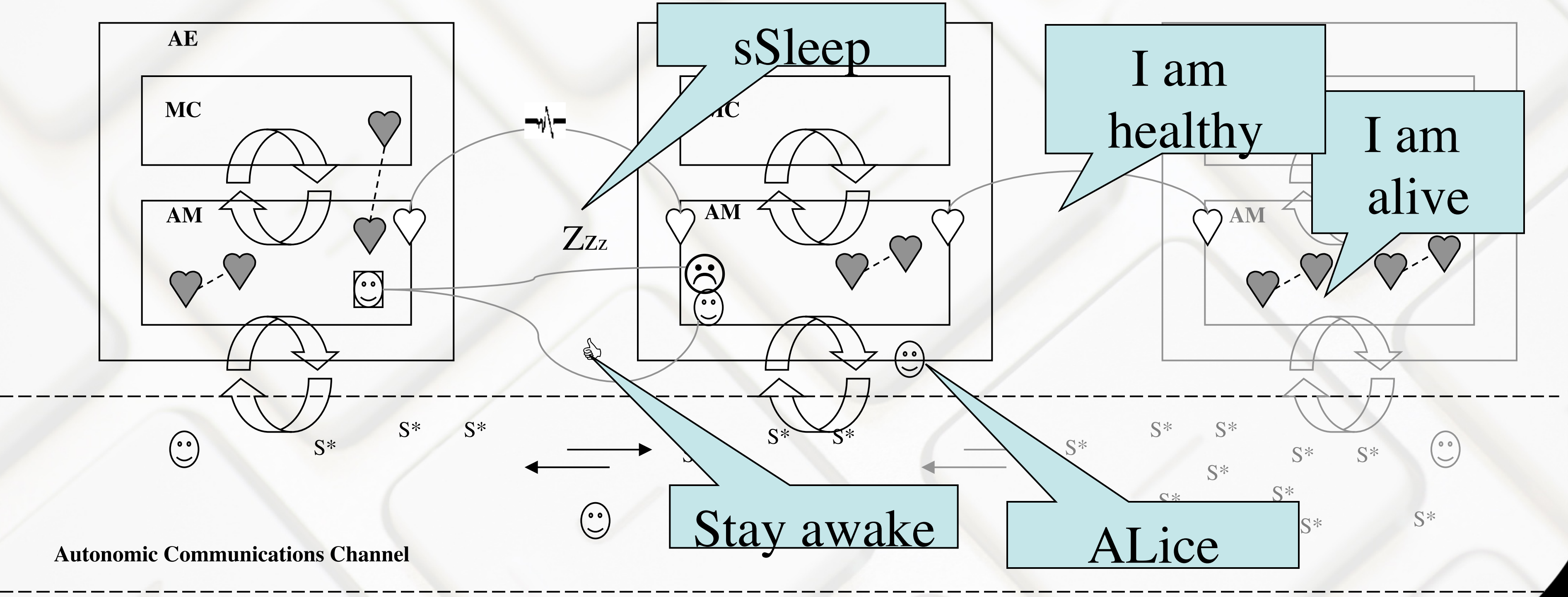
Fight or Flight

sympathetic (SyNS)

Rest and Digest

parasympathetic (PaNS)

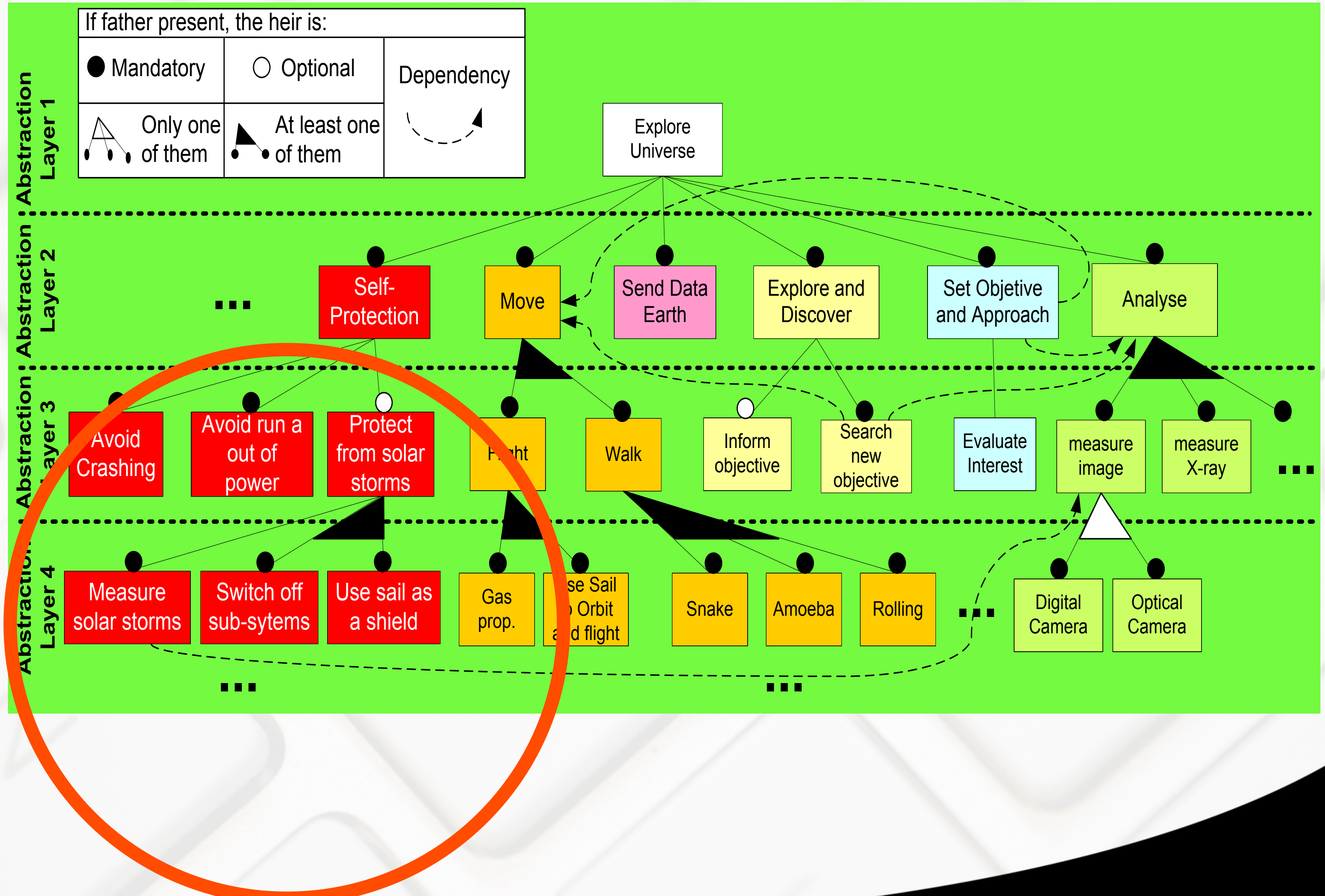
The Autonomic Environment



ECS Contributions

1. Formal Methods
2. Autonomic Computing
3. Software Product Lines
4. Automatic Code Generation

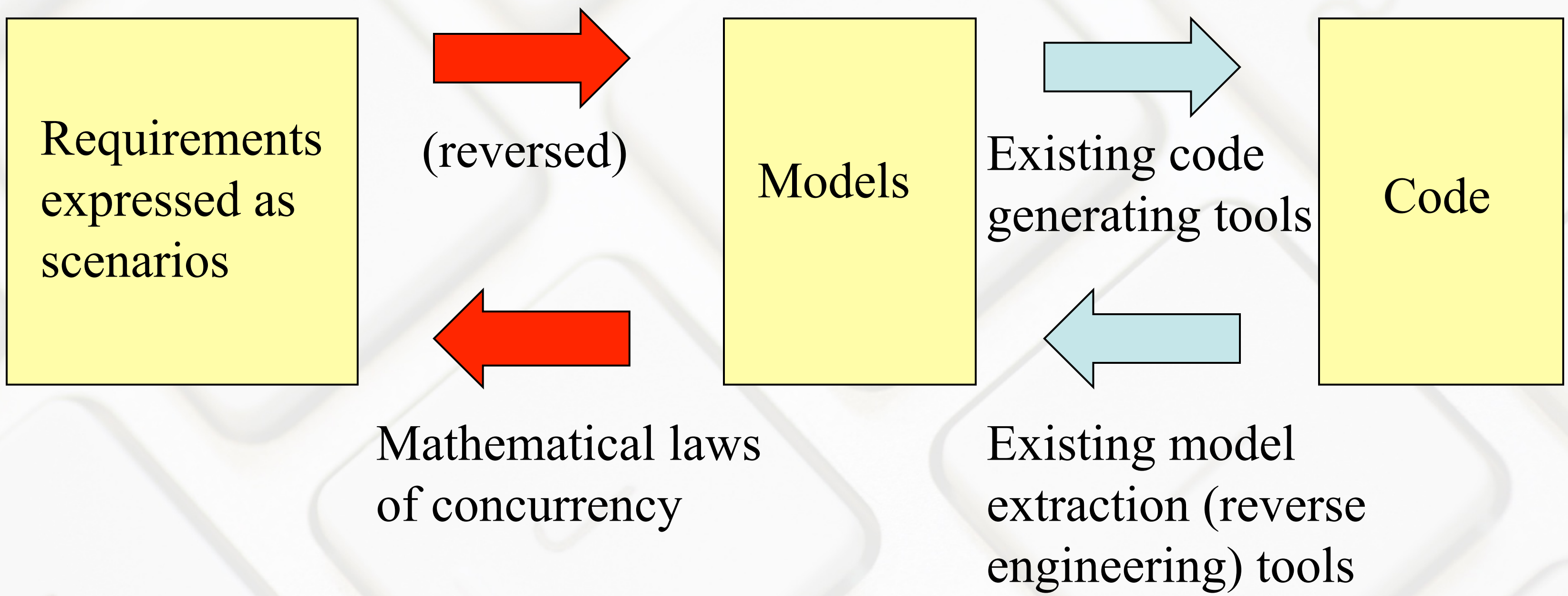
Feature Model



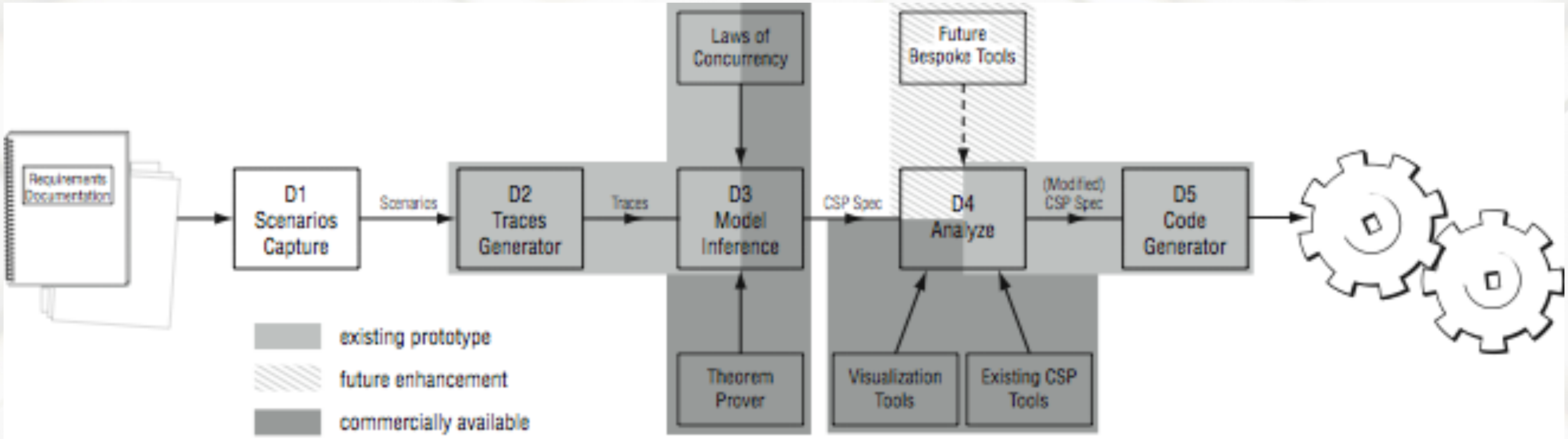
ECS Contributions

1. Formal Methods
2. Autonomic Computing
3. Software Product Lines
4. Automatic Code Generation

Requirements to Design to Code (R2D2C)



Current Status



Benefits of the Method

- Automation of entire development process;
- Significant increase in quality;
- Ability to do formal proof on properties of implementations;
- Ability to do formal proof of correctness;
- Automated means for requirements analysis;
- Guaranteed correspondence between requirements and their implementation as code.

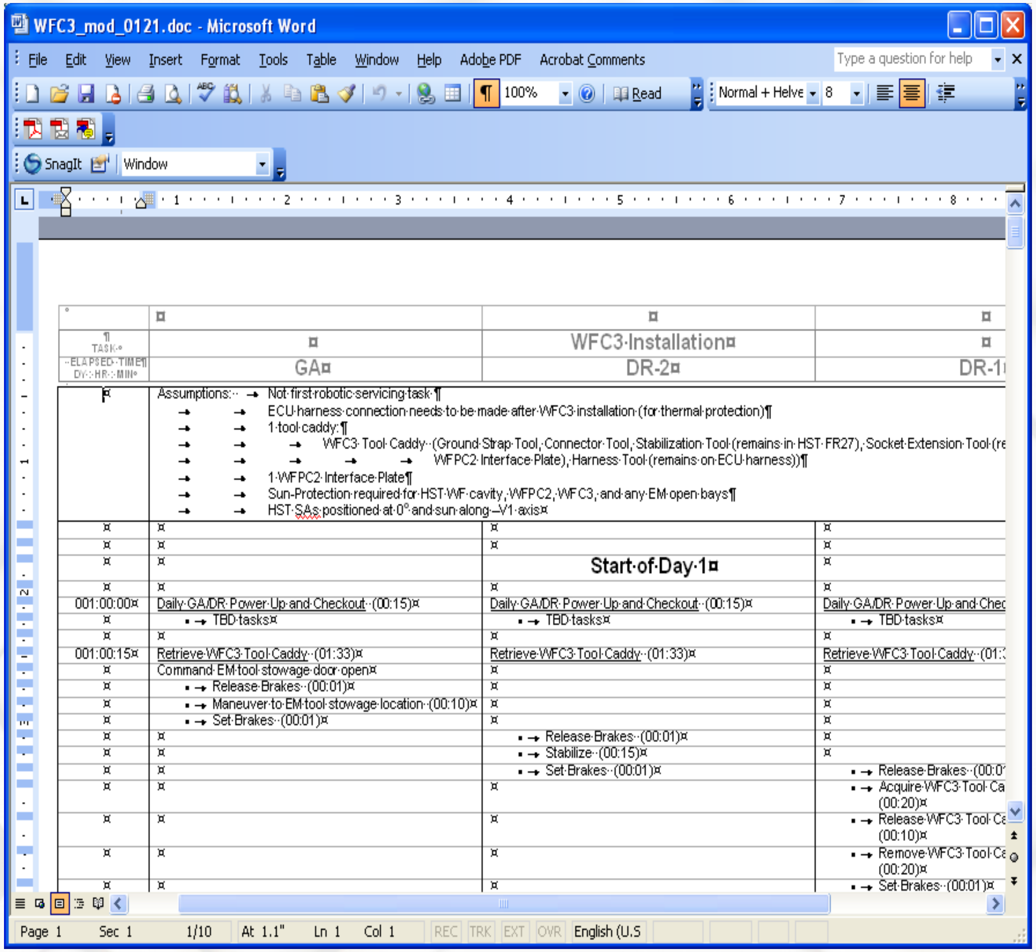
Applications

- End-to-end automatic code generation of provably correct systems;
- Automatic reimplementation after any requirements change;
- Exploiting re-use across platforms;
- Reverse engineering legacy systems to a mathematically sound model;
- Analysis and documentation of existing systems (e.g., expert systems);
- Re-engineering of legacy systems to a provably correct new implementation.

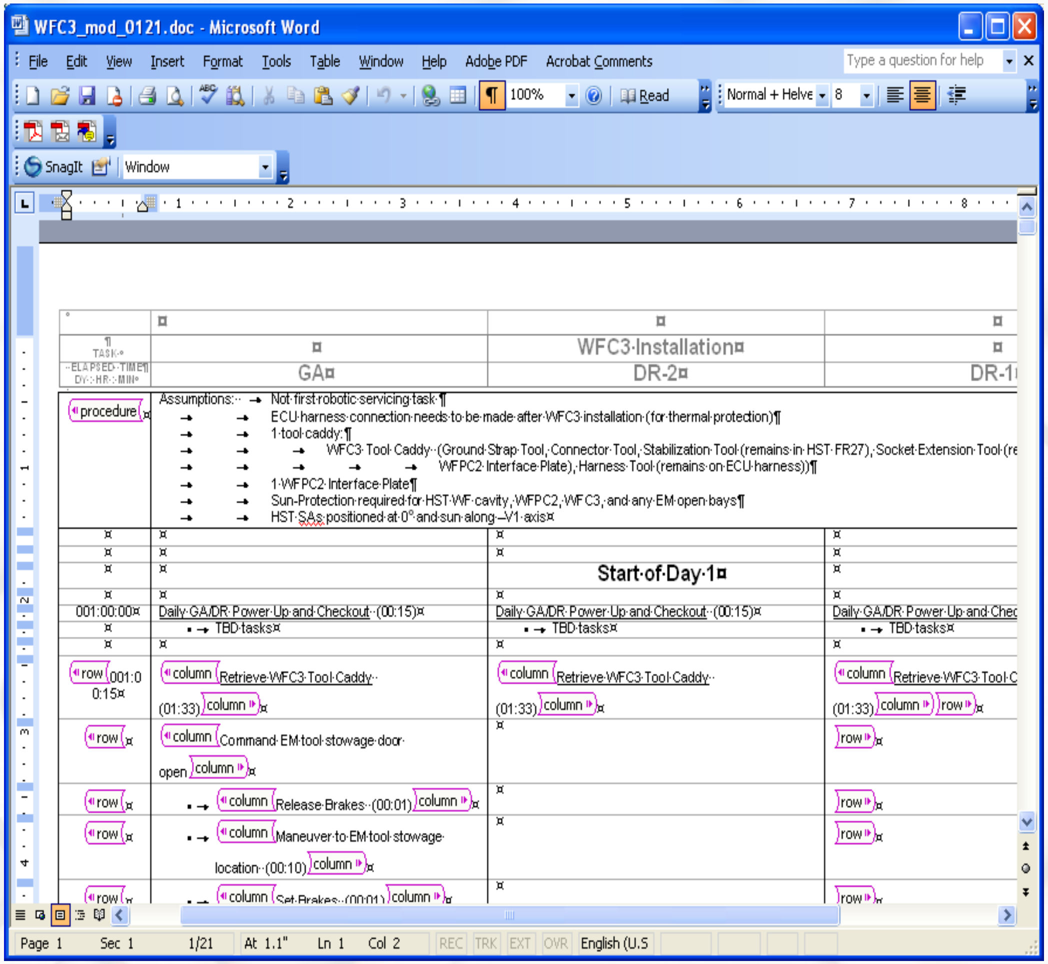
Domains (to date)

- Agent Based Systems;
- Wireless Sensor Networks ;
- ANTS;
- Verification of Robotic Procedures (cf. Hubble Space Telescope Robotic Servicing Mission).

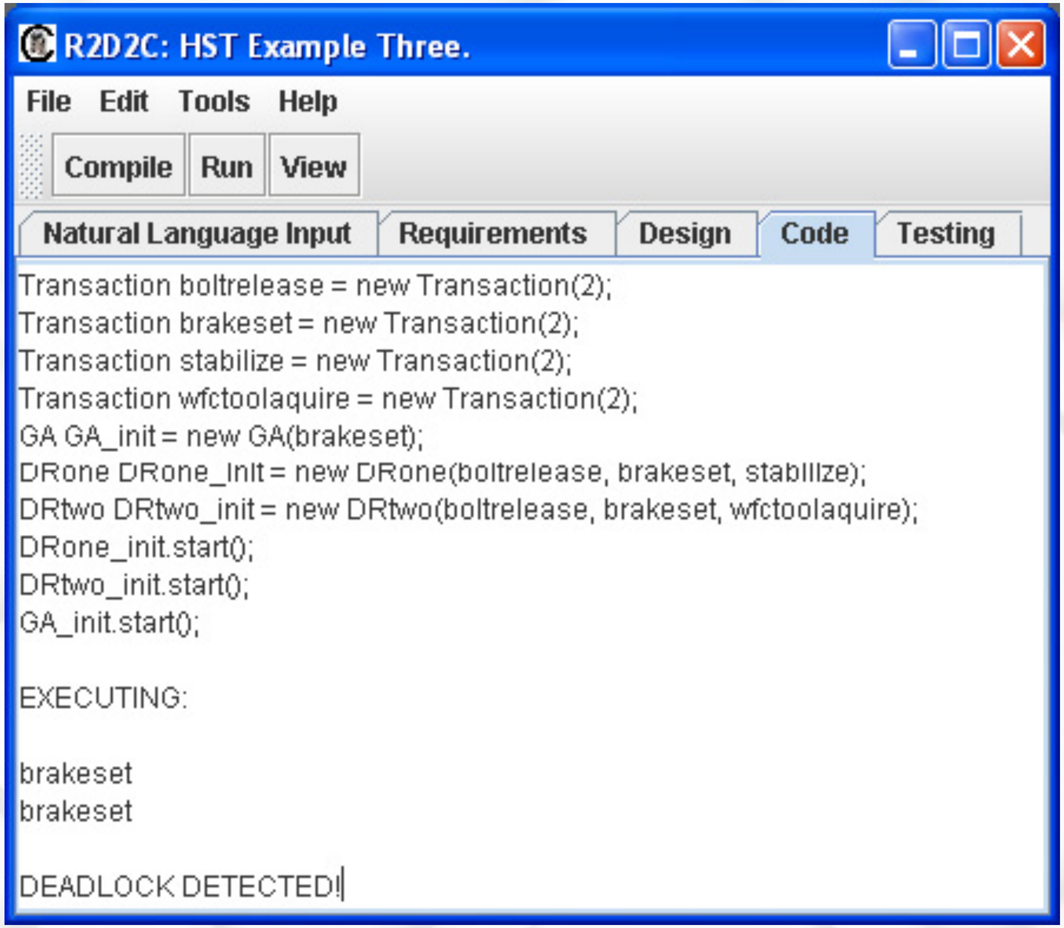
HRSM Procedures

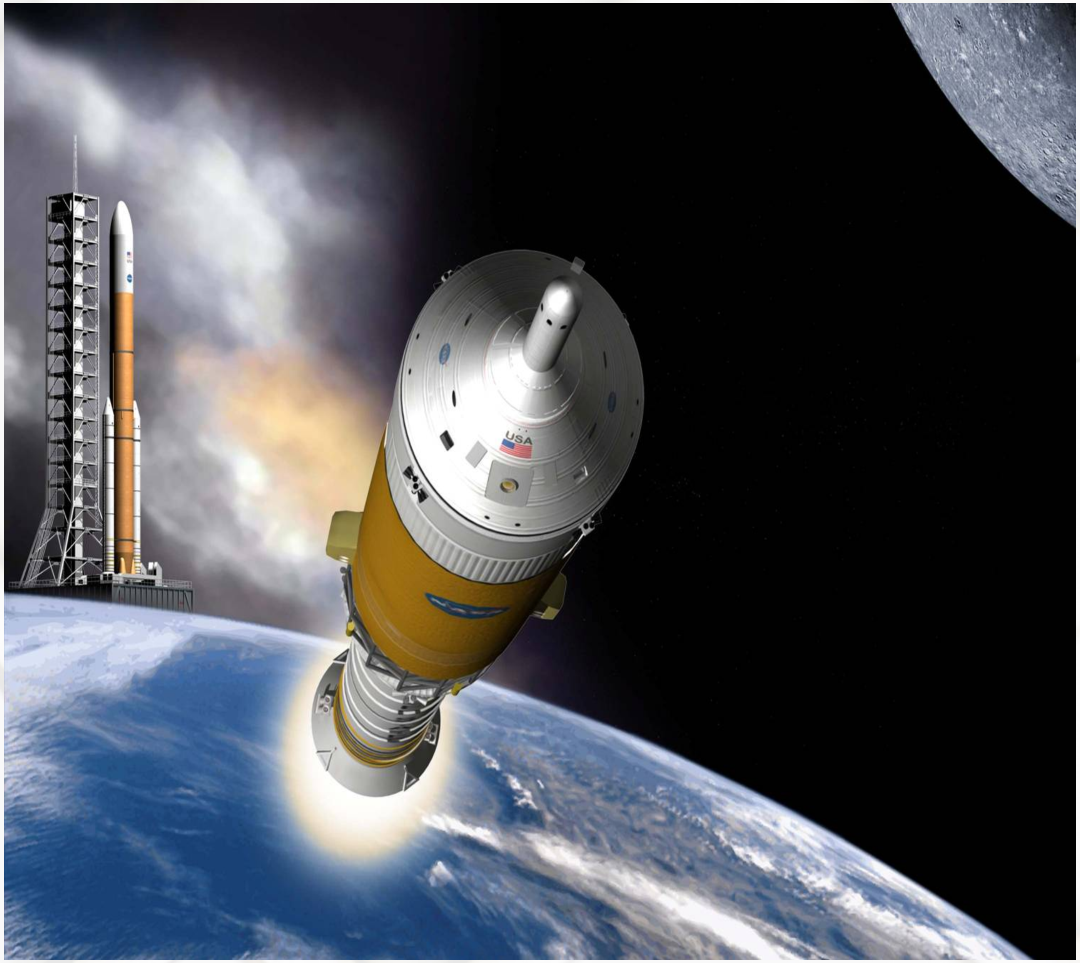


HRSM Procedures

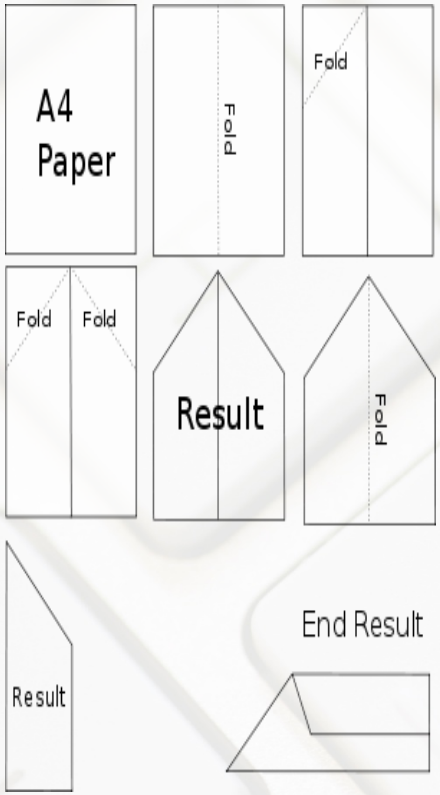



R2D2C: HST Example One.
File Edit Tools Help
Compile Run View
Natural Language Input
Requirements
Design
Code
Testing
channel brakerelease, brakeset, stabilize, wfctoolaquire : T ;
GA = brakeset ! 0 -> GA ;
DRone = brakeset ? x -> brakerelease ! 0 -> stabilize ! 0 -> brakeset ! 0 -> DRone ;
DRtwo = brakeset ? x -> wfctoolaquire ! 0 -> brakeset ! 0 -> DRtwo ;
System = DRone [| {| |} |] DRtwo [| {| |} |] GA ;

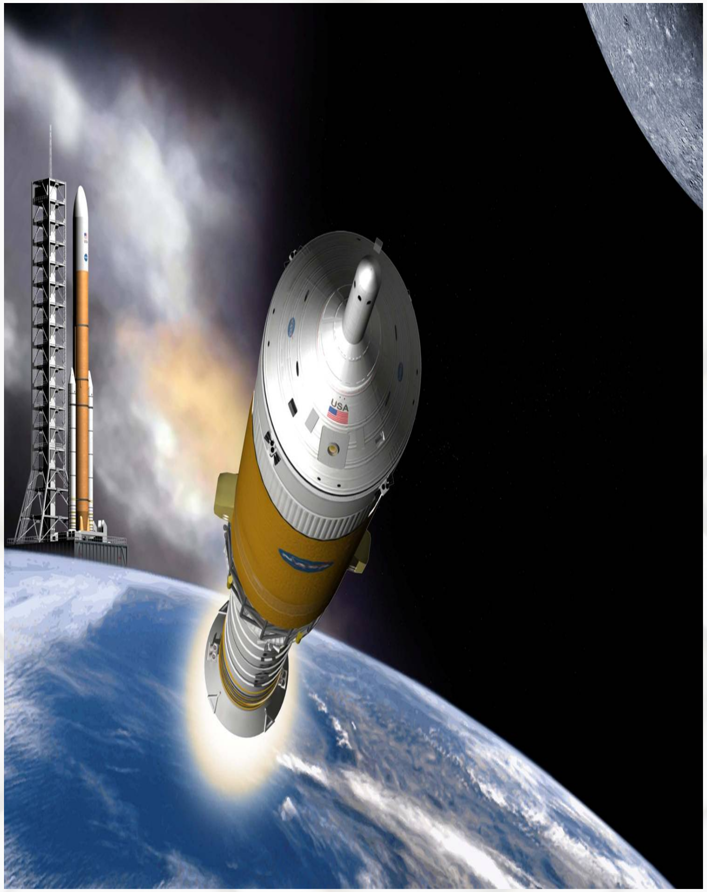

R2D2C: HST Example Three.
File Edit Tools Help
Compile Run View
Natural Language Input
Requirements
Design
Code
Testing
Transaction boltrelease = new Transaction(2);
Transaction brakeset = new Transaction(2);
Transaction stabilize = new Transaction(2);
Transaction wfctoolaquire = new Transaction(2);
GA GA_init = new GA(brakeset);
DRone DRone_init = new DRone(boltrelease, brakeset, stabilize);
DRtwo DRtwo_init = new DRtwo(boltrelease, brakeset, wfctoolaquire);
DRone_init.start();
DRtwo_init.start();
GA_init.start();
EXECUTING:
brakeset
brakeset
DEADLOCK DETECTED!


USA

Caveat



$\not\Rightarrow$



Conclusions

- Software must evolve.
- There is a tension between reliability, predictability and cost and this need for evolution.
- There is a need for an Evolving Critical Systems research effort.
- Lero and others are driving that effort.

Any problem in computer science can be solved with another layer of indirection.

But that usually will create another problem.

David Wheeler

Thank You

http://www.lero.ie/ecs/whitepaper

science foundation ireland
fondúireacht eolaíochta éireann

HEA
Higher Education Authority
An tÚdarás um Ard-Oideachas

NDP
PLEAN FORBARTHA NÁISIÚNTA