PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

INF646 Métodos formales

2018-2

Presentación del curso

Profesor del curso:

Виктор Хлебников
(Viktor Khlebnikov)

Master of Science in Engineering,
Faculty of Applied Mathematics and Computer Science,
Moscow State University
M. V. Lomonosov (MGU), 1975

Trabajo en industria: 1975 - 1992

Pontificia Universidad Católica del Perú,
Departamento de Ingeniería, Ingeniería Informática, 1993 -

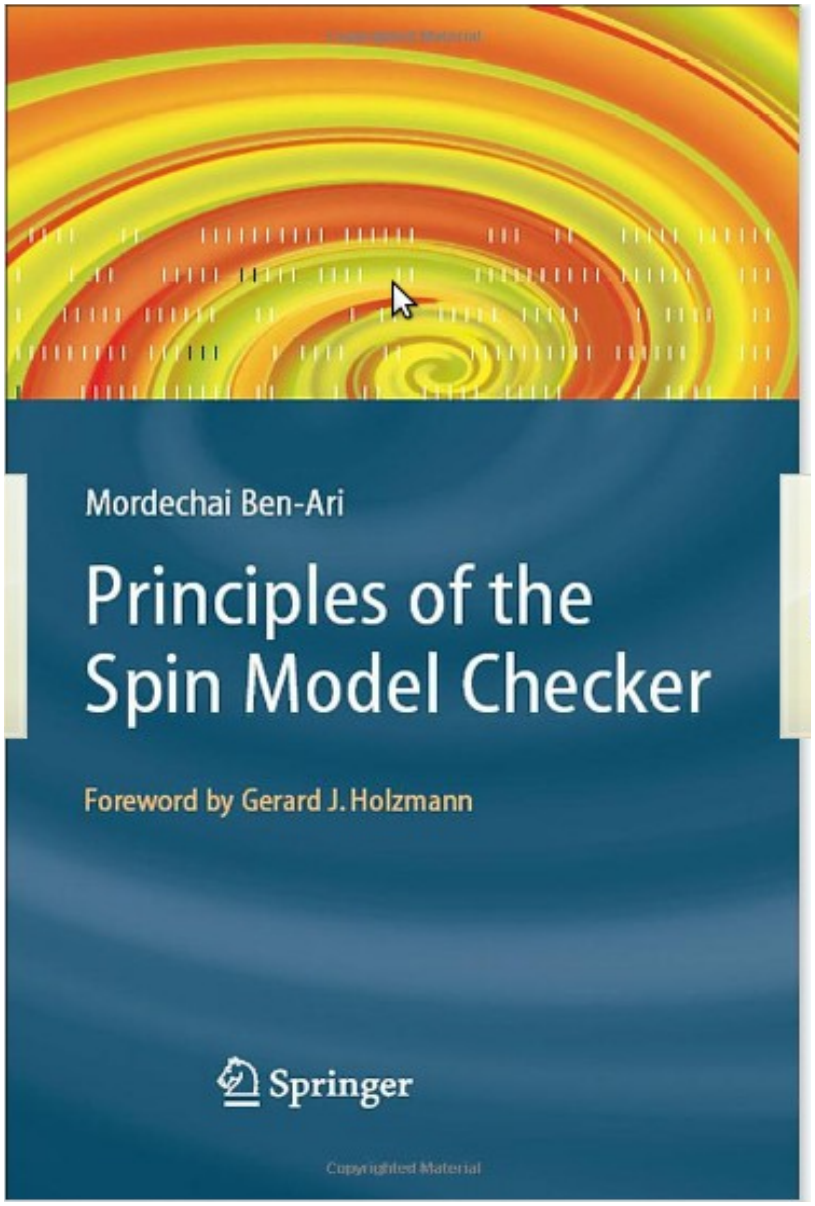
Pabellón V, 2do piso, oficina 6, teléfono: 6262000, anexo 4808
e-mail: vkhlebn@pucp.edu.pe
Skype: vkhlebn


dedmaxopka.livejournal.com © dedmaxopka@gmail.com

Objetivos del curso:

- aprender a describir modelos de software usando el lenguaje PROMELA;
- aprender a verificar el modelo con SPIN;
- manejar modelos de programas concurrentes.

Bibliografía del curso



Mordechai Ben-Ari

Principles of the Spin Model Checker

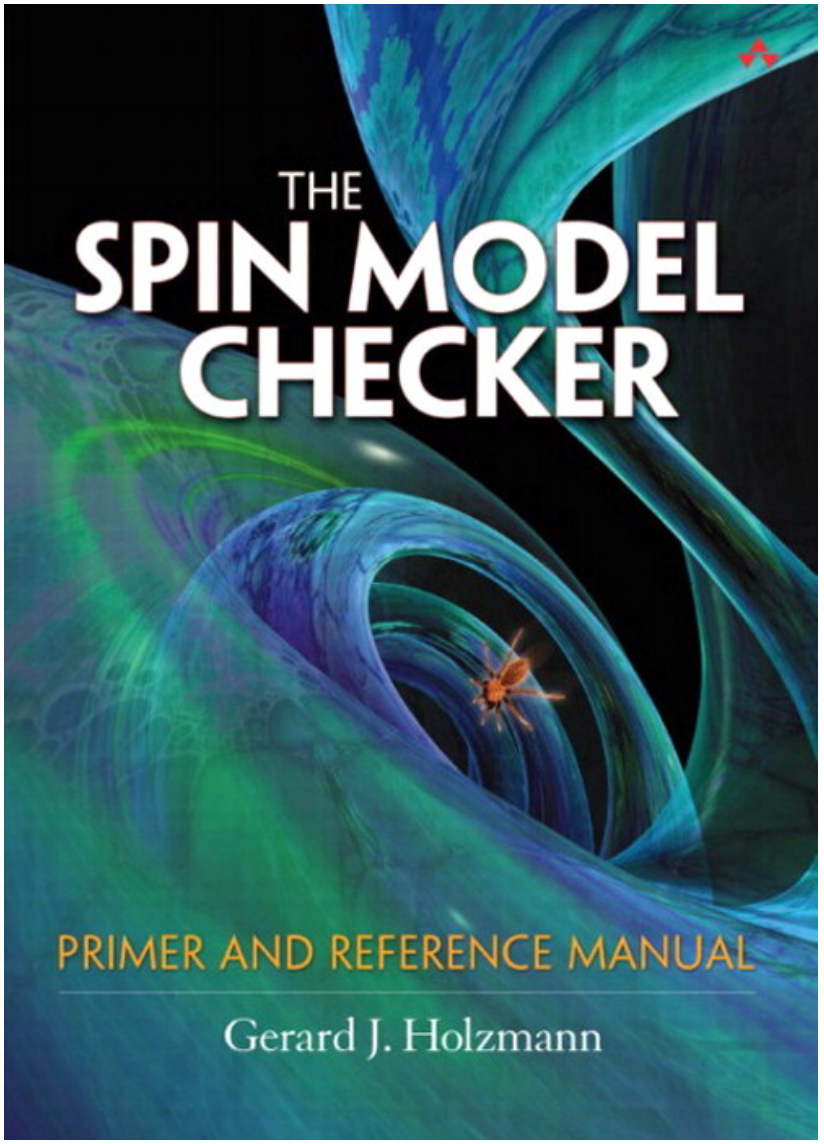
Springer, 2008

ISBN: 978-1-84628-769-5

QA 76.76.V47 B38
(Bib. Complejo de Innovación Académica)

[PSMC]

http://www.springer.com/computer/swe/book/978-1-84628-769-5
Supplementary material (zip, 38 kB)
978-1-84628-769-5-additional material.zip



Gerard J. Holzmann

The Spin Model Checker: Primer and Reference Manual

Addison-Wesley, 2004

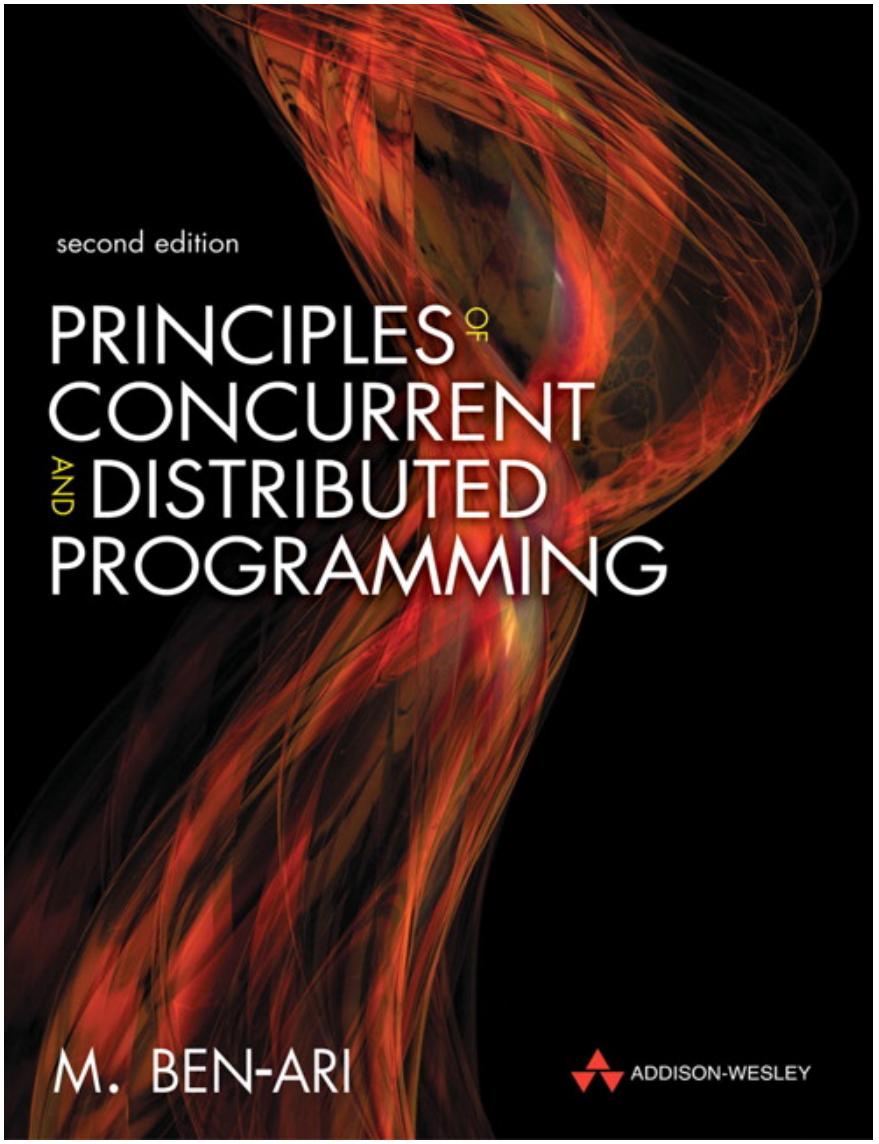
ISBN: 0-321-22862-6

QA 76.76.V47 H71
(Bib. Complejo de Innovación Académica)

[TSMC]

http://spinroot.com/spin/Doc/Book_extras/index.html
examples.tar.gz

Libros de consulta



Mordechai Ben-Ari

Principles of Concurrent and Distributed Programming, 2E

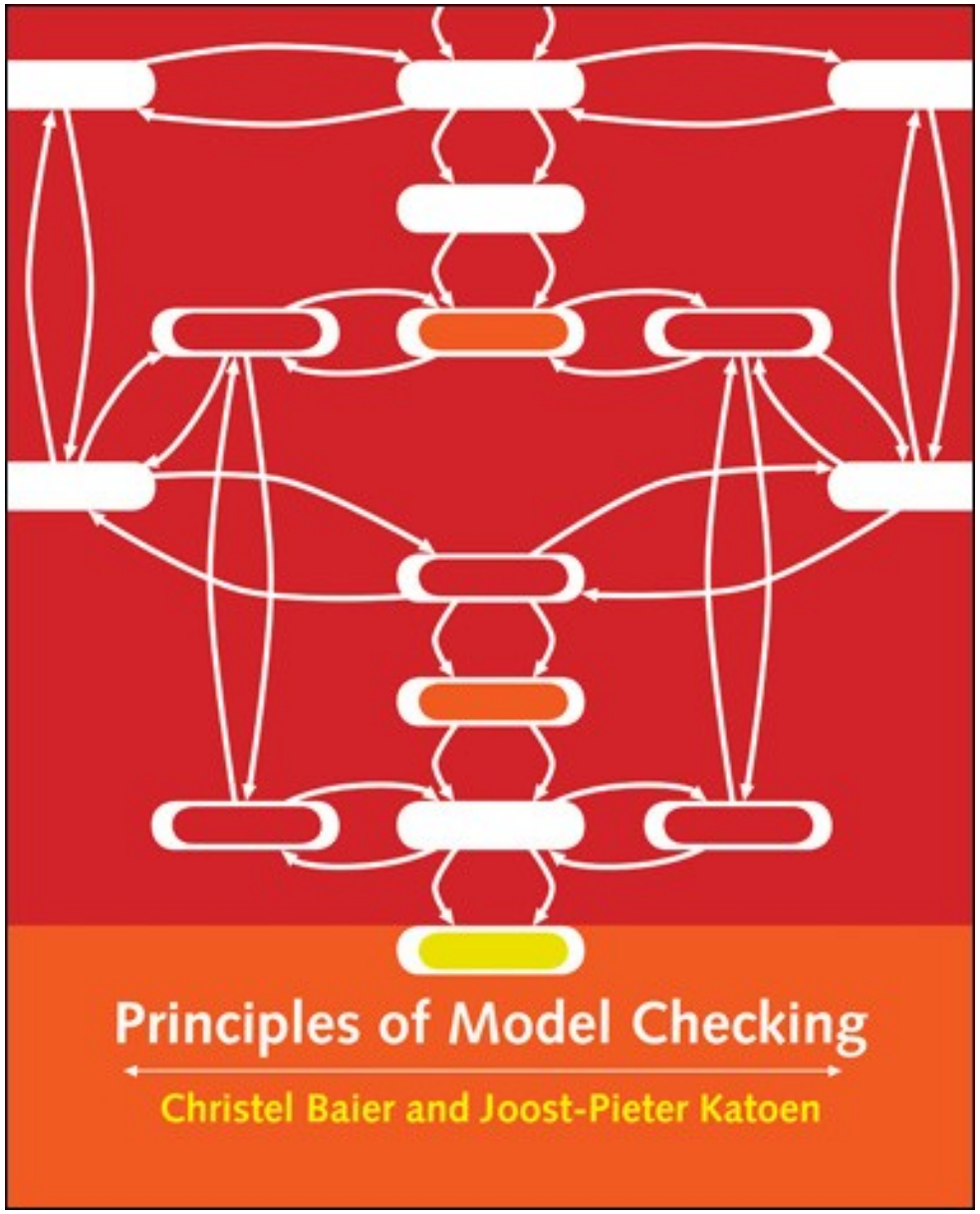
Addison-Wesley, 2006

ISBN: 0-32131-283-X

QA 76.9.D5 B38 2006
(Bib. Complejo de Innovación Académica)

[PCDP]

http://www.pearsoned.co.uk/highereducation/resources/Ben-AriPrinciplesofConcurrentandDistributedProgramming2e/
http://media.pearsoncmg.com/intl/ema/ema_uk_he_ben-ari_concprog_2/1405835532_prog_1-0.zip



Christel Baier and
Joost-Pieter Katoen

Principles of Model Checking

MIT Press, 2008

ISBN: 0-262-02649-X

QA 76.76.V47 B16
(Bib. Complejo de
Innovación Académica)

[PMC]

Formal Methods

From Wikipedia:

In computer science and software engineering, ***formal methods*** are a particular kind of mathematically based techniques for the

- specification,
- development and
- verification of software and hardware systems.

Formal Methods

From Wikipedia:

The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design.

Formal Methods

Formal methods are one of the "highly recommended" verification techniques for software development of safety-critical systems according to, e.g., the best practices standard of the IEC (International Electro-technical Commission) and standards of the ESA (European Space Agency).

PMC, p.7

Formal Methods

The resulting report of an investigation by the FAA (Federal Aviation Authority) and NASA about the use of formal methods concludes that

Formal methods should be part of the education of every computer scientist and software engineer, just as the appropriate branch of applied maths is a necessary part of the education of all other engineers.

PMC, p.7

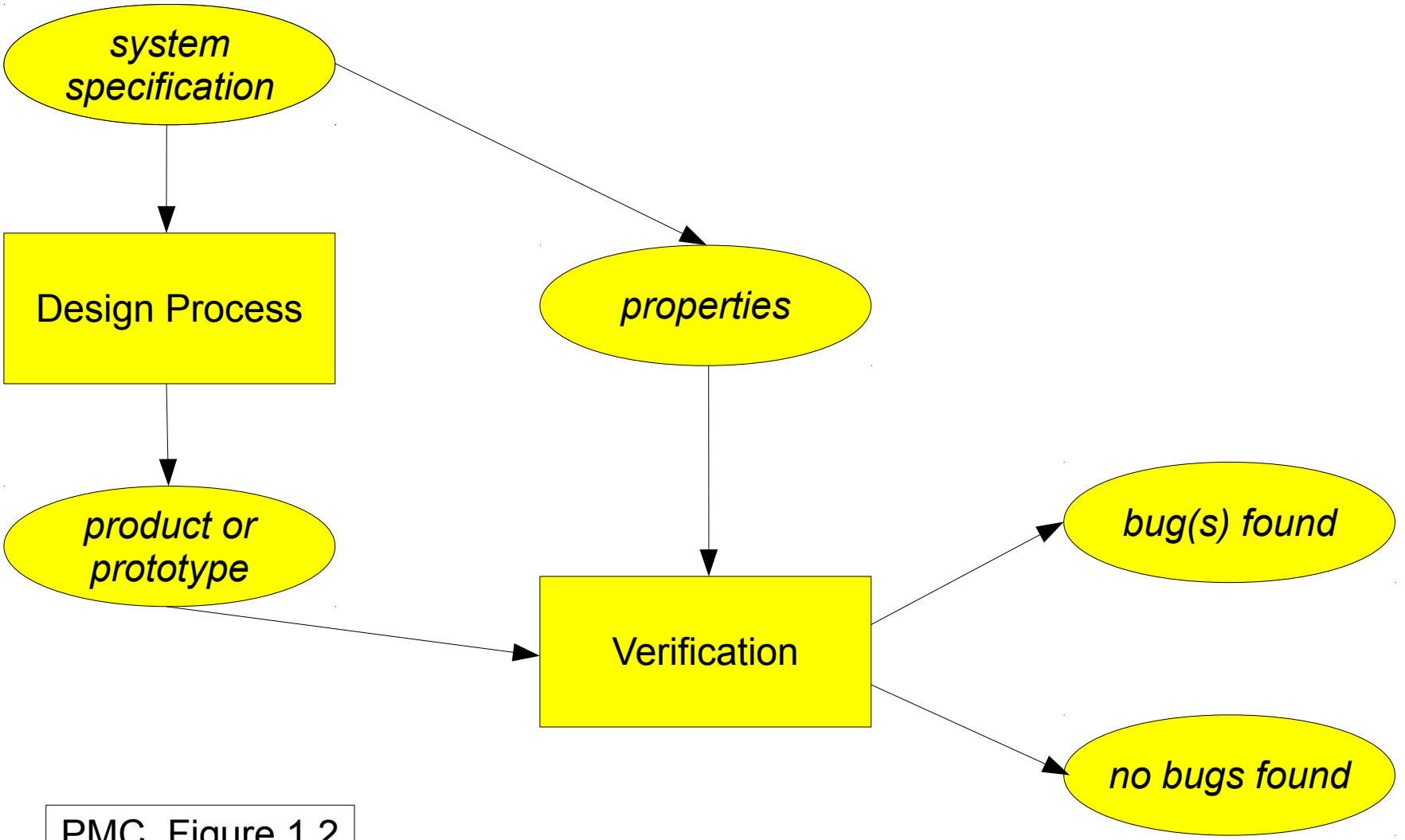
Formal Specification

From Wikipedia:

In computer science, ***formal specifications*** are mathematically based techniques whose purpose are to help with the implementation of systems and software.

Formal specifications describe ***what*** the system should do, not ***how*** the system should do it.

A Posteriori System Verification



PMC, Figure 1.2

Model Checking

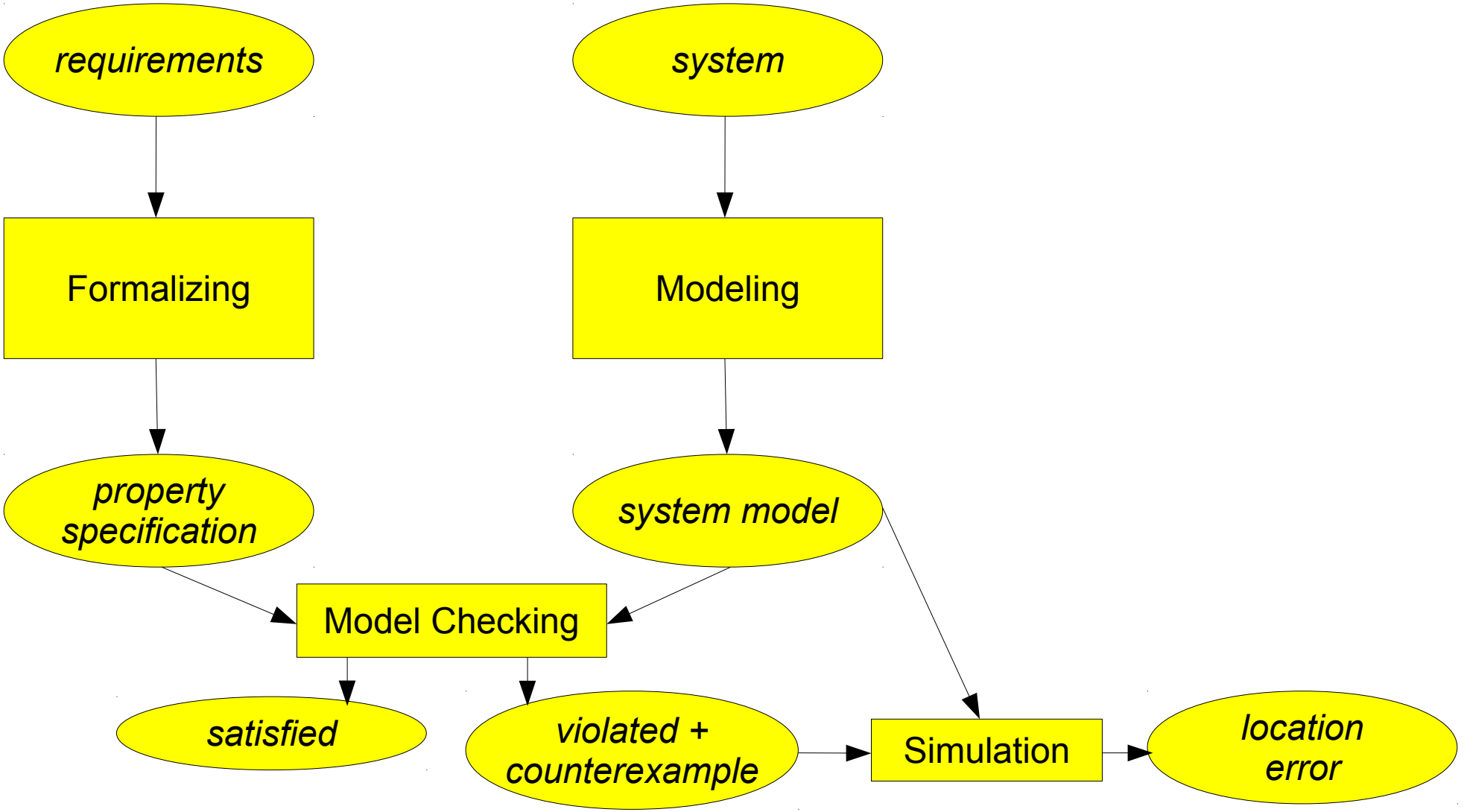
From Wikipedia:

In computer science, ***model checking*** or ***property checking*** refers to the following problem:

Given a model of a system, exhaustively and automatically check whether this model meets a given specification.

Typically, one has hardware or software systems in mind, whereas the specification contains safety requirements such as the absence of deadlocks and similar critical states that can cause the system to crash. Model checkingis a technique for automatically verifying correctness properties of *finite-state* systems.

A Model-Checking Approach



PMC, Figure 1.4

Principles of Model Checking

Any verification using model-based techniques is only as good as the model of the system.

Model checking is a verification technique that explores all possible system states in brute-force manner.

PMC, p.8

Principles of Model Checking

Model checking is an automated technique that,
given a finite-state model of a system and
a formal property,
systematically checks
whether this property holds for (a given state in)
that model.

PMC, p.11

SPIN =

*S*imple *P*romela *In*terpreter

Promela =

*Pro*cess *Me*ta-*La*nguage

TSMC, p.7,8

Inspiring Applications of Spin

Flood Control

Three examples of inspiring applications of Spin in the last few years include the verification of the control algorithms for the new flood control barrier built in the late nineties near Rotterdam in the Netherlands. The verification work was carried out by the Dutch firm CMG (Computer Management Group) in collaboration with the Formal Methods group at the University of Twente.

http://spinroot.com/spin/success.html

La **Maeslantkering** o **barrera de Maeslant** es una barrera contra la marejada ciclónica ubicada a la entrada del puerto de Róterdam (Países Bajos). Fue construida entre 1991 y 1997 como etapa final del Plan Delta. It is one of largest moving structures on Earth, rivalling the Green Bank Telescope in the United States and the Bagger 288 excavator in Germany.

El 9 de noviembre de 2007, la compuerta fue cerrada por primera vez desde su construcción en la década de 1990 para atender la emergencia que constituyó el avance de la tormenta "Tilo", que levantó, al menos al día 9, olas de hasta 3 metros de altura, provocando evacuaciones en Gran Bretaña y Países Bajos y alertas en Alemania. Para cerrar la barrera se tardó unas 2 horas.

https://en.wikipedia.org/wiki/Maeslantkering

The challenge

Because Rotterdam is a major port, it was critical to ensure that the Maeslant barrier was closed only when absolutely necessary and as briefly as possible. A closed barrier restricts shipping traffic, costing millions of euros. The barrier also had to be extremely reliable, with an error rate of no more than one flood every 10,000 years. To achieve this degree of reliability, an erroneous decision to leave the barrier open during a storm could not be made more than once every 100,000 times.

No room for error

Careful analysis determined that manual control of the barrier would limit its reliability. It is scientifically proven that a human being fails once every 1,000 times when making a decision. The Ministry of Infrastructure and the Environment decided that it was safer to let a computer make the decisions to close the Maeslant and Hartel barriers.

https://www.cgi.com/sites/default/files/pdf/rijkswaterstaat.pdf

The solution

CGI accepted the challenge to build a system to control the barriers on a fixed-price and fixed-time basis. Our experts developed and implemented BOS (an acronym for the system's Dutch name Beslis & Ondersteunend Systeem). The decision and support system decides whether to close the barriers in a storm based on weather readings from nearby weather stations and buoys.
BOS uses this data to calculate expected water levels near Rotterdam and Dordrecht every 10 minutes. When it recognizes a threat of flooding, it implements a series of required precautions and eventually starts the process to close the storm surge barriers independently, if necessary.

Mission-critical reliability requirement

The extremely low failure rate demanded of the BOS is very difficult to attain. In PC software, for example, a failure occurs almost every day. BOS had to be 36 million times more reliable, and this degree of reliability could not be proven by tests. It would take at least 2,000 years to do so. The high reliability requirements of the BOS placed it in the same category as nuclear power stations and the space shuttle.

https://www.cgi.com/sites/default/files/pdf/rijkswaterstaat.pdf

The result

Conventional development methods were inadequate to build a system like BOS. Instead, risk management had to be central to the development process right from the start. A well-structured, risk management-based development process with advanced methods and techniques was used.

More than 2,500 pages of specified technical design requirements were drafted, and more than 400,000 lines of code were written (in C++ and consists of 200,000 lines of code for the operational system and 250,000 lines of code for the simulation systems) to support the system's two fault-tolerant computers.

Because of the risks and costs involved, a very strict quality system was formulated specifically for the project.

As a result, the BOS project is the only one in the Netherlands to achieve ISO 9001 certification.

https://www.cgi.com/sites/default/files/pdf/rijkswaterstaat.pdf

The result (cont'd)

Further, techniques to ensure reliability were used, such as fault tree analysis, structured testing, an automatic test "environment" with periodical regression tests, design and code inspections, software reliability engineering, fault tolerance and many others.

The result is a system that experiences less than 10 minutes of downtime per year.

Further, reliability per barrier operation is higher than 99,995 percent.

Fully autonomous and uninfluenced by most conditions, the Maeslant barrier is sometimes referred to as the largest robot in the world.

BOS has also attained the highest Safety Integrity Level based on international safety standard IEC 61508.

https://www.cgi.com/sites/default/files/pdf/rijkswaterstaat.pdf

Inspiring Applications of Spin

Call Processing

Logic verification of the call processing software for a commercial data and phone switch, the PathStar switch that was designed and built at Lucent Technologies. The application was based on model extraction from the full and unmodified ANSI-C code of the implementation, which was checked for compliance with a group of roughly 20 class-5 features formalized in linear temporal logic (e.g., call waiting, conference calling, etc.). A cluster of 16 CPUs was used to perform the verifications overnight, every day for a period of several months before the switch was marketed. Perhaps the largest application of software model checking to date.

http://spinroot.com/spin/success.html

Inspiring Applications of Spin

Mission Critical Software

Selected algorithms for a number of space missions were verified with the Spin model checker.

The missions include

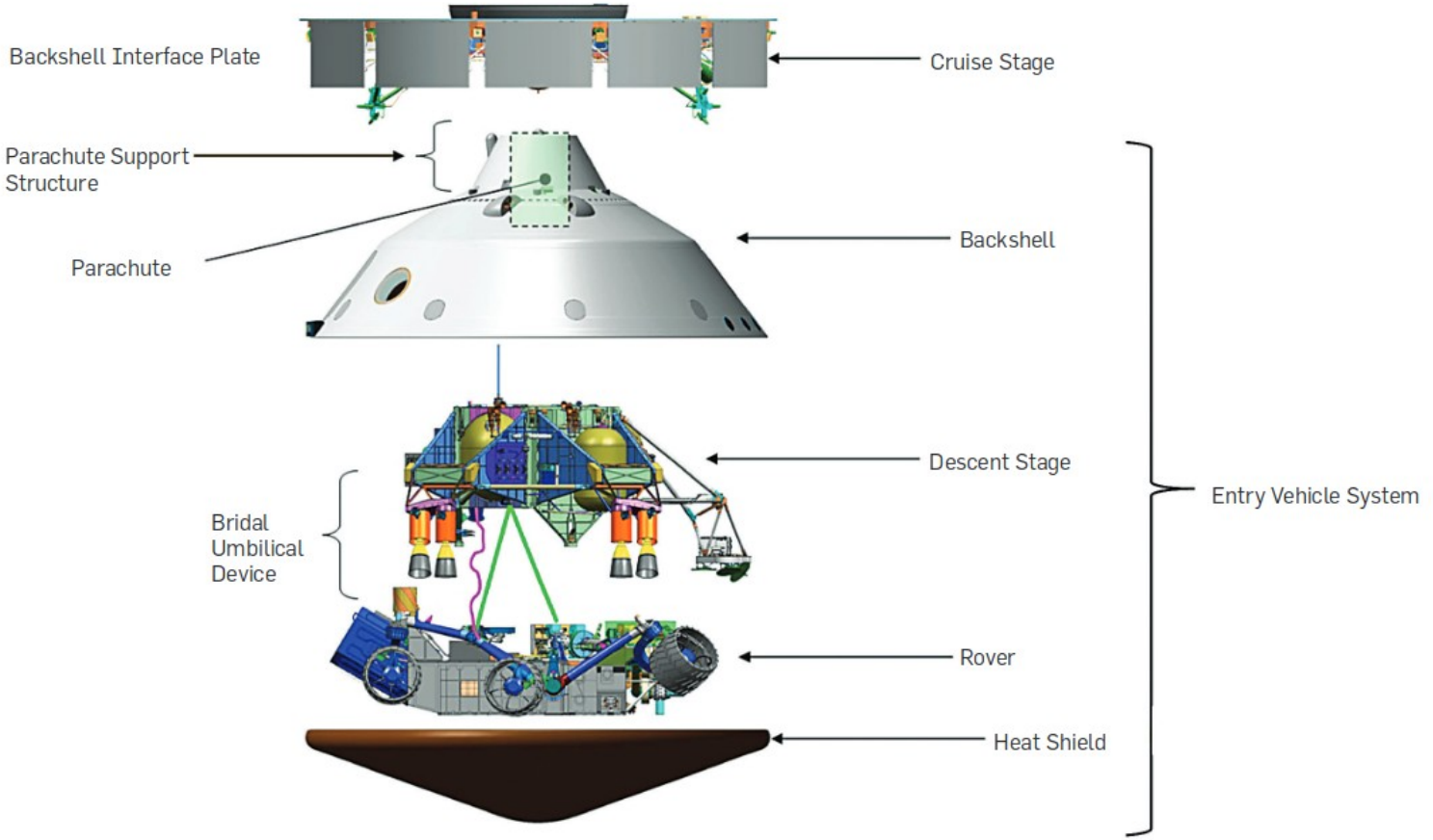
Mars Science Laboratory,
Deep Space 1,
Cassini,
the Mars Exploration Rovers,
Deep Impact, etc.

http://spinroot.com/spin/success.html

Inspiring Applications of Spin

Mars Science Laboratory

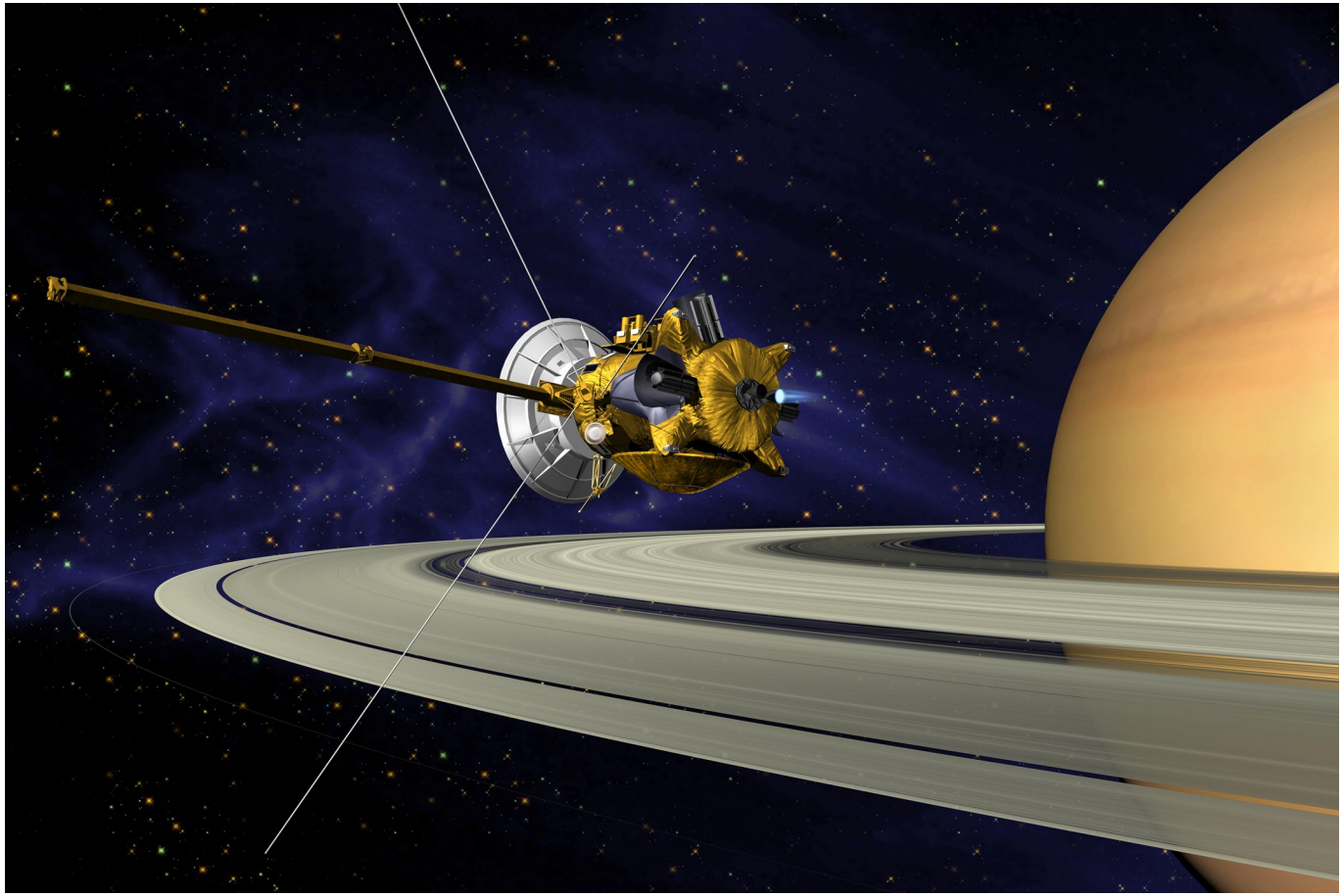
A description of the work done on MSL can be found here:
http://spinroot.com/gerard/pdf/cacm_2014.pdf



Inspiring Applications of Spin

Cassini

For Cassini, we verified the correct working of the handoff algorithms for the dual control CPUs (http://spinroot.com/gerard/pdf/nasa98.pdf).



Inspiring Applications of Spin

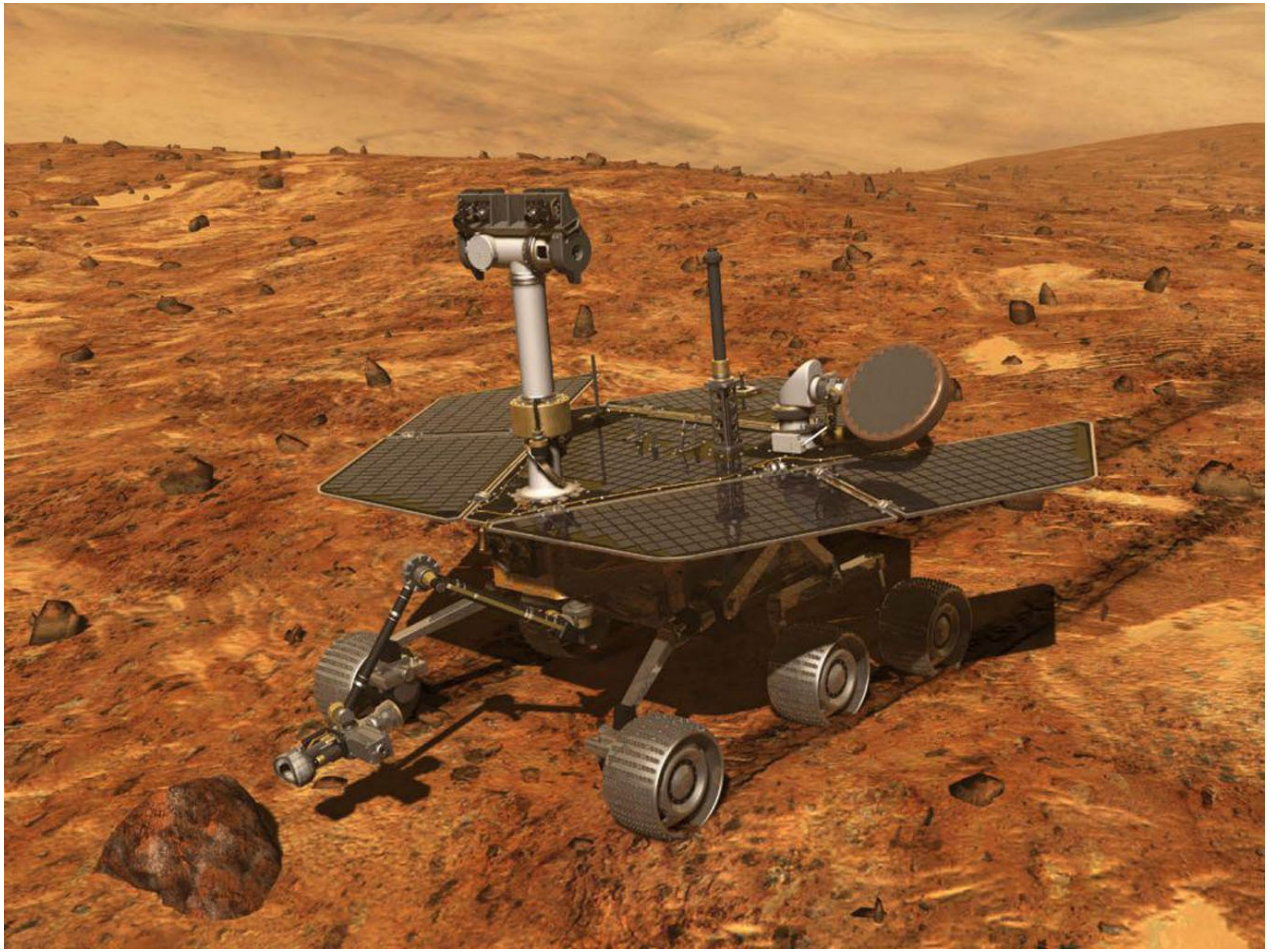
Deep Space 1

For Deep Space 1, researchers at Ames Research Center verified some key algorithms and reported their findings in a technical report (http://spinroot.com/spin/Doc/rax.pdf), and the post-mortem analysis in (http://spinroot.com/spin/Doc/remote-agent-ieee.pdf). Later, at JPL a separate verification of the software used on this mission was also done (http://spinroot.com/gerard/pdf/gluck_holz.pdf).

Inspiring Applications of Spin

Mars Exploration Rovers

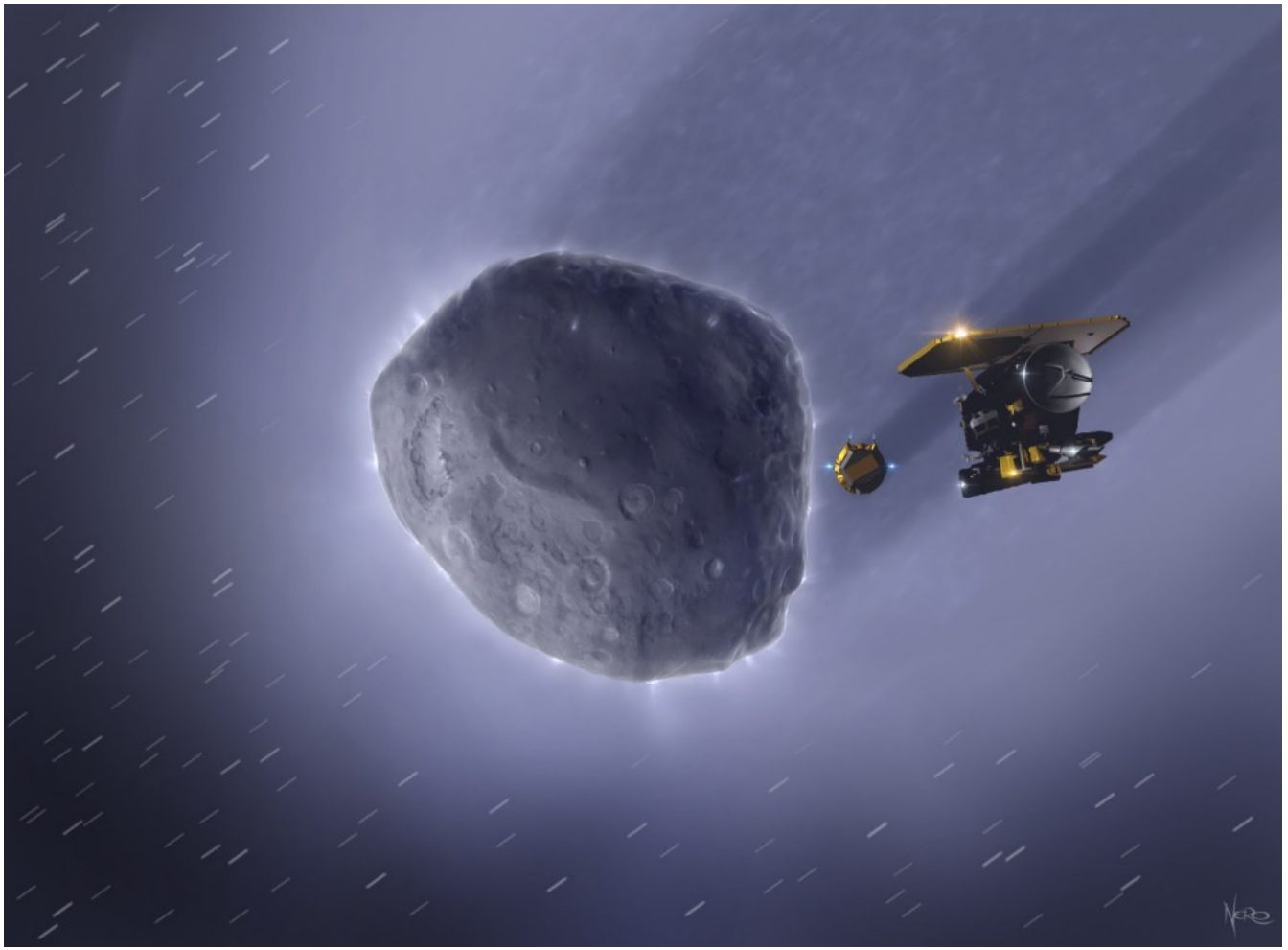
For the Mars Exploration Rovers we verified the correct working of the resource arbiter that manages the use of all motors on the rovers (http://spinroot.com/gerard/pdf/spin04.pdf).



Inspiring Applications of Spin

Deep Impact

More recently, for Deep Impact we verified aspects of the flash file system module that had shown problems before the encounter with the comet took place.



Inspiring Applications of Spin

Verification of medical device transmission protocols

Spin was used for about ten years in the verification of international standards that are used worldwide. This work started with the IQPS project at Eindhoven University, and continued at the University of Groningen, both in The Netherlands. Standards that were influenced by the Spin verification work include Firewire IEEE 1394.1 (used in many PCs), and ISO/IEEE 11073-20601 for medical devices.