



***University of Amsterdam***  
*Theory of Computer Science*

---

From Software Crisis to Informational  
Money

---

J.A. Bergstra

J.A. Bergstra

section Theory of Computer Science  
Faculty of Science  
University of Amsterdam

Science Park 904  
1098 XH Amsterdam  
the Netherlands

tel. +31 20 525.7591  
e-mail: [J.A.Bergstra@uva.nl](mailto:J.A.Bergstra@uva.nl)

# From Software Crisis to Informational Money

Jan A. Bergstra

Informatics Institute, University of Amsterdam  
email: `j.a.bergstra@uva.nl`

## 1 Introduction

I met Hans van Vliet in 1982 when I moved from Leiden to the Mathematical Centre (now CWI).<sup>1</sup> Together with the late Jaco de Bakker, Paul Klint, and Jan Heering we constituted a fairly heterogeneous community worried about computer software, each in our own particular manner and style. In those years it became increasingly popular to think in terms of a software crisis and scientific style research was assumed to be a critical ingredient on the path towards the solution of that crisis. These five individuals maintained entirely different approaches, and each of us thought that “he got it right”, alas for the other four, so to say. Apart from this ideological fragmentation our coexistence was entirely peaceful and often amusing.

## 2 Different styles and approaches

Paul Klint’s approach was, and still is, to act as a software architect and (chief) programmer and to perform research and engineering in a co-evolutionary style. In his view research needs an application, and new software (applications, nowadays simply app’s) needs research, and software tools for software construction allow that marriage in an ideal way. Jan Heering was fairly skeptical, interested in all forms of research, and very supportive and instrumental for that research as well, but he believed that practice always moves its own ways and that the influence of research is fairly limited.

Jaco de Bakker maintained that one must know precisely what those programs mean in order to deal with the chaos that surrounds this so-called software crisis. He turned that view into a reputable branch of research almost singlehandedly. I thought myself that specification and formalization are key to the solution of programming problems, and years later I am still working along such lines.<sup>2</sup>

---

<sup>1</sup>This text was printed on pages 9–13 in “Software Engineering Tales”, the Liber Amicorum for Hans van Vliet edited by Patricia Lago and Remco de Boer for the occasion of Hans’ retirement on November 28 of 2014.

<sup>2</sup>For instance, adapting [7] by working with an error value as is usual in Abstract Datatypes, in [6] we use algebraic specifications in order to formalize equational reasoning in elementary mathematics. The teaching of elementary mathematics (“rekenen-wiskunde”) shows signs of having its own crisis world wide, and in [3] we suggest “rekenen-informatica” (formalization and specification in disguise) as a potential way out.

Hans, however, felt that logical theory is not the core of the problem, neither is the meaning of programs, nor the underdevelopment of programming skills. He maintained and still maintains the following views (as expressed in my own words):

- A software crisis manifests itself as a crisis of products with problems and that measuring these problems, or their absence in quantitative terms for instance with software metrics, is critical, however problematic and even unrewarding the theory of that sort of thing may appear to be.
- If products have problems these problems arise from production processes which are problematic as well. A top down approach to the structure of software production processes is useful. That approach must necessarily be far more general than what can be concluded or observed from one's own hands on programming experience.
- Software production admits a classification in terms of product categories. Once a product class has been taken in focus dedicated software architectures become accessible. There is no way around a systematic though "application dependent" investigation of software architecture and the systematic application of the results of that investigation.

It seems that history has proven Hans right, at least to the extent that the directions that he has been following are at this moment still very active and lively. When looking at the research output of Hans van Vliet as displayed by Google Scholar I noticed that by far the most visible part of his work has been written in this millennium. He was consistently very active, writing a remarkable sequence of well-cited papers throughout the last 15 years: I mention for instance [1, 9, 10].

### 3 The power of software

What we failed to see so clearly back in 1982 is how critically important software would become for almost all aspects of economic and scientific life. Nowadays the software crisis has different forms and shapes. Security problems abound and most of those are software problems. The ability to deal with security issues has become a decisive factor for obtaining and maintaining a competitive edge in software manufacturing. Software projects are still running out of budget at an alarming rate but this is now seen as a problem for local government. As software companies have proven to be extremely profitable the idea that "software is in crisis" at large has proven mistaken. Software legacy issues are still all over the place and geological layers of technically outdated but steadily operating software accumulate.

The revolution of these years is the phantastic increase in data storage capacity. The idea of a database management system carefully designed to collect, store, and process precisely the data that an organizations needs seems to have become outdated. Instead software is written that helps an organization to find its way in "big" amounts of data. This data analytics technology constitutes the software side of data science, which is the friendly face of an upcoming data crisis. indeed the impact of outdated legacy data may be even more problematic than that of legacy software.

An important manifestation of the power of software is open source software. By being a descendant of Linux, Android proves beyond any possible doubt that open source software production processes cannot be neglected by commercial software manufacturing companies anymore. Now one might think that open source impact is somehow confined to operating systems, compilers and other systems software but I believe that that is not true and that the real force of open source software production has yet to emerge.

## 4 Informational money

An interesting case of open source software development is the Bitcoin system. Since the (still) anonymous software authoring agent Satoshi Nakamoto wrote about Bitcoin in [11] and released the first open source implementation of it, the development of Bitcoin has been amazingly fast. There is no doubt that Bitcoin is primarily an achievement in terms of software architecture. Today's commenters are unanimous in the verdict that Bitcoin may disappear (and so Bitcoin holders with high expectations may become deeply disappointed), but blockchain technology has come to stay. Bitcoin uses the blockchain idea in its simplest form, but much more of that sort is to come. Blockchain technology may become vital for the solution of the data crisis by combining universal accessibility with best possible security in flexible degrees of anonymity.

The open source background of Bitcoin strengthens its acceptance in certain milieus. On the one hand that is seen as a weakness, as those circles are remote from the world of international banking. On the other hand that forces a communal effort to produce, maintain, and use competitive software for distributed financial (or other) transactions. It is questionable that a Bitcoin-like system could have emerged as the outcome of conventional commercial software production. This sort of question merits further research.

It seems that via Bitcoin open source software may influence our perception of money. In [4, 8] we proposed the notion of informational money, and, with Bitcoin in mind, the more extreme notion of exclusively informational money which does away with conventional concepts of ownership. In [2] I argue that Bitcoin provides a remarkable platform for the implementation of Islamic Finance.

However remote Islamic Finance may seem from a conventional point of view, in [5] we have argued that as a design option for a money, its built-in limitations may be somehow understood by comparing it with the development of new program notations. A programming style is often characterized by the features it does not favor which a corresponding program notation for that reason does not (or only reluctantly) offer on purpose.

## 5 Conclusion

Software and in particular software architecture, the primary focus of the research of Hans, are much more important than merely by having practical and instrumental value, these are conceptual tools for the future. The concept of money is only one of many classical concepts that may be in need of revision with novel software architectures in mind.

## References

- [1] Per Olof Bengtsson, Nico Lassing, Jan Bosch, and Hans van Vliet. Architecture-level modifiability analysis (ALMA), *J. of Ssystems and Software* vol. 69 (1-2), pages 129-147 (2004).
- [2] Jan A. Bergstra. Bitcoin and Islamic Finance. University of Amsterdam, Informatics Institute, Report TCS1406, April 2014, <http://www.science.uva.nl/pub/programming-research/tcsreports/TCS1406.pdf> (2014).
- [3] Jan A. Bergstra, Inge Bethke, and Alban Ponse Rekenen-Informatica. University of Amsterdam, Section Theory of Computer Science, Report TCS1412. <https://ivi.fnwi.uva.nl/tcs/pub/tcsreports/TCS1412.pdf> (2014).
- [4] Jan A. Bergstra and Karl de Leeuw. Bitcoin and Beyond: Exclusively Informational Money. [arXiv:1304.4758v2](https://arxiv.org/abs/1304.4758v2) [cs.CY] (2013).
- [5] J.A. Bergstra and C.A. Middelburg. Preliminaries to an investigation of reduced product set finance. *JKAU: Islamic Economics*, 24(1):175–210 (2011).
- [6] Jan A. Bergstra and Alban Ponse Division by zero in common meadows [arXiv:1406.6878](https://arxiv.org/abs/1406.6878) [math.RA] (2014).
- [7] J.A. Bergstra and J.V. Tucker. The rational numbers as an abstract data type. *Journal of the ACM*, 54 (2), Article 7 (2007).
- [8] Jan A. Bergstra and Peter Weijland. Bitcoin: a money-like informational commodity. [arXiv:1402.4778](https://arxiv.org/abs/1402.4778), (2014).
- [9] Jaap Gordijn, Hans Akkermans, and Hans van Vliet, Business modelling is not process modelling. In: *Conceptual modeling for e-business and the web*, pages 40–51, Springer, (2000).
- [10] Philippe Kruchten, Patricia Lago, and Hans van Vliet, Building up and reasoning about architectural knowledge. QoSA 2006, LNCS 4214, pages 43-58 (2006).
- [11] Satoshi Nakamoto. Bitcoin: a peer-to-peer electronic cash system. <http://Bitcoin.org/Bitcoin.pdf> (2008).

## Electronic Reports Series of section Theory of Computer Science

---

Within this series the following reports appeared.

- [TCS1413] J.A. Bergstra and I. Bethke, *Note on Paraconsistency on the Logic of Fractions*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1412] J.A. Bergstra, I. Bethke, and A. Ponse, *Rekenen-Informatica*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1411] J.A. Bergstra, *Bitcoin: not a Currency-like Informational Commodity*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1409v2] J.A. Bergstra and A. Ponse, *Three Datatype Defining Rewrite Systems for Datatypes of Integers each extending a Datatype of Naturals (version 2)*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1410] J.A. Bergstra and A. Ponse, *Division by Zero in Common Meadows*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1407v3] J.A. Bergstra, *Four Complete Datatype Defining Rewrite Systems for an Abstract Datatype of Natural Numbers (version 3)*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1409] J.A. Bergstra and A. Ponse, *Three Datatype Defining Rewrite Systems for Datatypes of Integers each extending a Datatype of Naturals*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1406v3] J.A. Bergstra, *Bitcoin and Islamic Finance (version 3)*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1407v2] J.A. Bergstra, *Four Complete Datatype Defining Rewrite Systems for an Abstract Datatype of Natural Numbers (version 2)*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1408] J.A. Bergstra, *Bitcoin: Informational Money en het Einde van Gewoon Geld*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1407] J.A. Bergstra, *Four Complete Datatype Defining Rewrite Systems for an Abstract Datatype of Natural Numbers*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1406v2] J.A. Bergstra, *Bitcoin and Islamic Finance (version 2)*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1406] J.A. Bergstra, *Bitcoin and Islamic Finance*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1405] J.A. Bergstra, *Rekenen in een Conservatieve Schrapwet Weide*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1404] J.A. Bergstra, *Division by Zero and Abstract Data Types*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1403] J.A. Bergstra, I. Bethke, and A. Ponse, *Equations for Formally Real Meadows*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1402] J.A. Bergstra and W.P. Weijland, *Bitcoin, a Money-like Informational Commodity*, section Theory of Computer Science - University of Amsterdam, 2014.
- [TCS1401] J.A. Bergstra, *Bitcoin, een "money-like informational commodity"*, section Theory of Computer Science - University of Amsterdam, 2014.

- [TCS1301] B. Diertens, *The Refined Function-Behaviour-Structure Framework*, section Theory of Computer Science - University of Amsterdam, 2013.
- [TCS1202] B. Diertens, *From Functions to Object-Orientation by Abstraction*, section Theory of Computer Science - University of Amsterdam, 2012.
- [TCS1201] B. Diertens, *Concurrent Models for Object Execution*, section Theory of Computer Science - University of Amsterdam, 2012.
- [TCS1102] B. Diertens, *Communicating Concurrent Functions*, section Theory of Computer Science - University of Amsterdam, 2011.
- [TCS1101] B. Diertens, *Concurrent Models for Function Execution*, section Theory of Computer Science - University of Amsterdam, 2011.
- [TCS1001] B. Diertens, *On Object-Orientation*, section Theory of Computer Science - University of Amsterdam, 2010.

Within former series (PRG) the following reports appeared.

- [PRG0914] J.A. Bergstra and C.A. Middelburg, *Autosolvability of Halting Problem Instances for Instruction Sequences*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0913] J.A. Bergstra and C.A. Middelburg, *Functional Units for Natural Numbers*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0912] J.A. Bergstra and C.A. Middelburg, *Instruction Sequence Processing Operators*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0911] J.A. Bergstra and C.A. Middelburg, *Partial Komori Fields and Imperative Komori Fields*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0910] J.A. Bergstra and C.A. Middelburg, *Indirect Jumps Improve Instruction Sequence Performance*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0909] J.A. Bergstra and C.A. Middelburg, *Arithmetical Meadows*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0908] B. Diertens, *Software Engineering with Process Algebra: Modelling Client / Server Architectures*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0907] J.A. Bergstra and C.A. Middelburg, *Inversive Meadows and Divisive Meadows*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0906] J.A. Bergstra and C.A. Middelburg, *Instruction Sequence Notations with Probabilistic Instructions*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0905] J.A. Bergstra and C.A. Middelburg, *A Protocol for Instruction Stream Processing*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0904] J.A. Bergstra and C.A. Middelburg, *A Process Calculus with Finitary Comprehended Terms*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0903] J.A. Bergstra and C.A. Middelburg, *Transmission Protocols for Instruction Streams*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0902] J.A. Bergstra and C.A. Middelburg, *Meadow Enriched ACP Process Algebras*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0901] J.A. Bergstra and C.A. Middelburg, *Timed Tuplix Calculus and the Wesseling and van den Berg Equation*, Programming Research Group - University of Amsterdam, 2009.
- [PRG0814] J.A. Bergstra and C.A. Middelburg, *Instruction Sequences for the Production of Processes*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0813] J.A. Bergstra and C.A. Middelburg, *On the Expressiveness of Single-Pass Instruction Sequences*, Programming Research Group - University of Amsterdam, 2008.



- [PRG0812] J.A. Bergstra and C.A. Middelburg, *Instruction Sequences and Non-uniform Complexity Theory*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0811] D. Staudt, *A Case Study in Software Engineering with PSF: A Domotics Application*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0810] J.A. Bergstra and C.A. Middelburg, *Thread Algebra for Poly-Threading*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0809] J.A. Bergstra and C.A. Middelburg, *Data Linkage Dynamics with Shedding*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0808] B. Diertens, *A Process Algebra Software Engineering Environment*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0807] J.A. Bergstra, S. Nolst Trenite, and M.B. van der Zwaag, *Tuplix Calculus Specifications of Financial Transfer Networks*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0806] J.A. Bergstra and C.A. Middelburg, *Data Linkage Algebra, Data Linkage Dynamics, and Priority Rewriting*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0805] J.A. Bergstra, S. Nolst Trenite, and M.B. van der Zwaag, *UvA Budget Allocatie Model*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0804] J.A. Bergstra and C.A. Middelburg, *Thread Algebra for Sequential Poly-Threading*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0803] J.A. Bergstra and C.A. Middelburg, *Thread Extraction for Polyadic Instruction Sequences*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0802] A. Barros and T. Hou, *A Constructive Version of AIP Revisited*, Programming Research Group - University of Amsterdam, 2008.
- [PRG0801] J.A. Bergstra and C.A. Middelburg, *Programming an Interpreter Using Molecular Dynamics*, Programming Research Group - University of Amsterdam, 2008.

The above reports and more are available through the website: [ivi.fnwi.uva.nl/tcs/](http://ivi.fnwi.uva.nl/tcs/)

## Electronic Report Series

---

section Theory of Computer Science  
Faculty of Science  
University of Amsterdam

Science Park 904  
1098 XG Amsterdam  
the Netherlands

[ivi.fnwi.uva.nl/tcs/](http://ivi.fnwi.uva.nl/tcs/)