Programming Methodology and Type Theory
How 40 years of uncompromising research made concrete the Chalmers futuristic vision into the Software Engineering paradigm of Programs correct by construction

Gérard Huet

Inria Paris Laboratory

Bengt Nordström Fest,
Chalmers University
April 28th 2016

Futuristic

When Chet Murthy left the Coq project to join IBM, he said:
Formal Methods are the way of the future ...

Futuristic

When Chet Murthy left the Coq project to join IBM, he said:
Formal Methods are the way of the future ...
and they always will be.

Irrelevance for commercial software development

Most ‘practitioners’ consider that software correctness is secondary and even harmful when time to market is the main issue. They consider formal methods are an un-necessary hindrance to fast code production, and reserved for archane debates by pompous academics.

Irrelevance for commercial software development

Most ‘practitioners’ consider that software correctness is secondary and even harmful when time to market is the main issue. They consider formal methods are an un-necessary hindrance to fast code production, and reserved for archane debates by pompous academics.
We must admit, our tools are not robust enough, and their wide dissemination is still too far away, to competitively design or even refactor a large software endeavor such as Word.

Irrelevance for commercial software development

Most ‘practitioners’ consider that software correctness is secondary and even harmful when time to market is the main issue. They consider formal methods are an un-necessary hindrance to fast code production, and reserved for archane debates by pompous academics.
We must admit, our tools are not robust enough, and their wide dissemination is still too far away, to competitively design or even refactor a large software endeavor such as Word.
But bugs have an enormous cost.

Irrelevance for commercial software development

Most ‘practitioners’ consider that software correctness is secondary and even harmful when time to market is the main issue. They consider formal methods are an un-necessary hindrance to fast code production, and reserved for archane debates by pompous academics.
We must admit, our tools are not robust enough, and their wide dissemination is still too far away, to competitively design or even refactor a large software endeavor such as Word.
But bugs have an enormous cost. Thus hardware design, safety-critical systems and security have progressively used formal methods in production software development.

Irrelevance for commercial software development

Most ‘practitioners’ consider that software correctness is secondary and even harmful when time to market is the main issue. They consider formal methods are an un-necessary hindrance to fast code production, and reserved for archane debates by pompous academics.
We must admit, our tools are not robust enough, and their wide dissemination is still too far away, to competitively design or even refactor a large software endeavor such as Word.
But bugs have an enormous cost. Thus hardware design, safety-critical systems and security have progressively used formal methods in production software development.
Note that our tools are now able to prove the correctness of the first program in Knuth’s Art of Computer Programming, and that’s not small achievement.

Relevance to serious programming

We used to think of λ-calculus as a theoretical formalism for computable partial functions. But:

Relevance to serious programming

We used to think of λ-calculus as a theoretical formalism for computable partial functions. But:

- Lisp

Relevance to serious programming

We used to think of λ-calculus as a theoretical formalism for computable partial functions. But:

- Lisp
- Iswim

Relevance to serious programming

We used to think of λ-calculus as a theoretical formalism for computable partial functions. But:

- Lisp
- Iswim
- ML

Relevance to serious programming

We used to think of λ-calculus as a theoretical formalism for computable partial functions. But:

- Lisp
- Iswim
- ML
- G-machine, LML

Relevance to serious programming

We used to think of λ-calculus as a theoretical formalism for computable partial functions. But:

- Lisp
- Iswim
- ML
- G-machine, LML
- Haskell

Relevance to serious programming

We used to think of λ-calculus as a theoretical formalism for computable partial functions. But:

- Lisp
- Iswim
- ML
- G-machine, LML
- Haskell

Nowadays, the banking business is taking seriously functional programming.

The software engineering scene in the 80’s

The software engineering scene in the 80's

- Programming methods

The software engineering scene in the 80's

- Programming methods
- Software Science

The software engineering scene in the 80's

- Programming methods
- Software Science
- Structured Programming

The software engineering scene in the 80's

- Programming methods
- Software Science
- Structured Programming
- Floyd, Hoare, Dijkstra, Abrial, etc.

The software engineering scene in the 80's

- Programming methods
- Software Science
- Structured Programming
- Floyd, Hoare, Dijkstra, Abrial, etc.
- Artificial Intelligence

The software engineering scene in the 80's

- Programming methods
- Software Science
- Structured Programming
- Floyd, Hoare, Dijkstra, Abrial, etc.
- Artificial Intelligence
- Abstract Datatypes and Algebraic Specifications

The software engineering scene in the 80's

- Programming methods
- Software Science
- Structured Programming
- Floyd, Hoare, Dijkstra, Abrial, etc.
- Artificial Intelligence
- Abstract Datatypes and Algebraic Specifications
- Object-oriented programming

The software engineering scene in the 80's

- Programming methods
- Software Science
- Structured Programming
- Floyd, Hoare, Dijkstra, Abrial, etc.
- Artificial Intelligence
- Abstract Datatypes and Algebraic Specifications
- Object-oriented programming
- Denotational semantics

Visionary efforts

Visionary efforts

- Automath

Visionary efforts

- Automath
- LCF

Visionary efforts

- Automath
- LCF
- Nuprl

Visionary efforts

- Automath
- LCF
- Nuprl
- Curry-Howard-de Bruijn isomorphism

Visionary efforts

- Automath
- LCF
- Nuprl
- Curry-Howard-de Bruijn isomorphism
- Type theory

Visionary efforts

- Automath
- LCF
- Nuprl
- Curry-Howard-de Bruijn isomorphism
- Type theory
- Correctness by construction

Programming Methodology at Chalmers

Programming Methodology at Chalmers

- It started in Midsummer 1979

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980
- Edited as "Intuitionistic Type Theory" studies in Proof Theory 1984

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980
- Edited as "Intuitionistic Type Theory" studies in Proof Theory 1984
- There was this intriguing series of summer workshops in wonderful places such as Marstrand and Båstad

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980
- Edited as "Intuitionistic Type Theory" studies in Proof Theory 1984
- There was this intriguing series of summer workshops in wonderful places such as Marstrand and Båstad
- It attracted researchers curious about this new trend

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980
- Edited as "Intuitionistic Type Theory" studies in Proof Theory 1984
- There was this intriguing series of summer workshops in wonderful places such as Marstrand and Båstad
- It attracted researchers curious about this new trend
- The types meeting organized by Gilles Kahn in 1984 in Sophia-Antipolis

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980
- Edited as "Intuitionistic Type Theory" studies in Proof Theory 1984
- There was this intriguing series of summer workshops in wonderful places such as Marstrand and Båstad
- It attracted researchers curious about this new trend
- The types meeting organized by Gilles Kahn in 1984 in Sophia-Antipolis
- Programming in Martin-Löf's Type Theory 1990

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980
- Edited as "Intuitionistic Type Theory" studies in Proof Theory 1984
- There was this intriguing series of summer workshops in wonderful places such as Marstrand and Båstad
- It attracted researchers curious about this new trend
- The types meeting organized by Gilles Kahn in 1984 in Sophia-Antipolis
- Programming in Martin-Löf's Type Theory 1990
- ESPRIT Basic Research Actions Logical Frameworks, then Types

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980
- Edited as "Intuitionistic Type Theory" studies in Proof Theory 1984
- There was this intriguing series of summer workshops in wonderful places such as Marstrand and Båstad
- It attracted researchers curious about this new trend
- The types meeting organized by Gilles Kahn in 1984 in Sophia-Antipolis
- Programming in Martin-Löf's Type Theory 1990
- ESPRIT Basic Research Actions Logical Frameworks, then Types
- BRA consistency with the uncompromizing attitude

Programming Methodology at Chalmers

- It started in Midsummer 1979
- Martin-Löf's lectures in Padua June 1980
- Edited as "Intuitionistic Type Theory" studies in Proof Theory 1984
- There was this intriguing series of summer workshops in wonderful places such as Marstrand and Båstad
- It attracted researchers curious about this new trend
- The types meeting organized by Gilles Kahn in 1984 in Sophia-Antipolis
- Programming in Martin-Löf's Type Theory 1990
- ESPRIT Basic Research Actions Logical Frameworks, then Types
- BRA consistency with the uncompromizing attitude
- Long term goals are solved by "doing things right", a commitment to quality

40 years of uncompromising research

40 years of uncompromising research

Persistency vs obstinacy vs stubbornness

40 years of uncompromising research

Persistency vs obstinacy vs stubbornness
Swedish qualities ?

40 years of uncompromising research

Persistency vs obstinacy vs stubbornness
Swedish qualities ?
Indulging in stereotypes and flirting with political incorrectness, again

40 years of uncompromising research

Persistency vs obstinacy vs stubbornness
Swedish qualities ?
Indulging in stereotypes and flirting with political incorrectness, again
Blacklisted for 20 years, and still number one on the hit list

Swedish bashing

Swedish bashing

hard-working

Swedish bashing

hard-working
methodic

Swedish bashing

hard-working
methodic
honest

Swedish bashing

hard-working
methodic
honest

Where once the Indian to the death
Chased pioneer and scout,
The Swede, with alcoholic breath,
Sets rows of cabbage out.

The art of the research team leader

The art of the research team leader

Administration is not Science

The art of the research team leader

Administration is not Science
Thus sane researchers avoid administration duties

The art of the research team leader

Administration is not Science
Thus sane researchers avoid administration duties
But research needs resources, and money is controled by Law

The art of the research team leader

Administration is not Science
Thus sane researchers avoid administration duties
But research needs resources, and money is controled by Law
Thus the research team leader must deal with legalities

The art of the research team leader

Administration is not Science
Thus sane researchers avoid administration duties
But research needs resources, and money is controled by Law
Thus the research team leader must deal with legalities
The research team leader is a facilitator and he makes administration transparent

The art of the research team leader

Administration is not Science
Thus sane researchers avoid administration duties
But research needs resources, and money is controled by Law
Thus the research team leader must deal with legalities
The research team leader is a facilitator and he makes administration transparent
For efficiency, he must learn the rules, and apply his scientific skills to remove obstacles in a legal way

The science of creative bureaucracy

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them
Know the rules better than the bureaucrat in charge of regulating your requests

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them

Know the rules better than the bureaucrat in charge of regulating your requests

Defeat the opponent using game theory, logic, and psychology

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them

Know the rules better than the bureaucrat in charge of regulating your requests

Defeat the opponent using game theory, logic, and psychology

Winning by putting the opponent on your side

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them
Know the rules better than the bureaucrat in charge of regulating your requests
Defeat the opponent using game theory, logic, and psychology
Winning by putting the opponent on your side
Frighten the bureaucrat using your natural endowments

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them
Know the rules better than the bureaucrat in charge of regulating your requests
Defeat the opponent using game theory, logic, and psychology
Winning by putting the opponent on your side
Frighten the bureaucrat using your natural endowments
Use logic to prove that obstacles may be bypassed

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them
Know the rules better than the bureaucrat in charge of regulating your requests
Defeat the opponent using game theory, logic, and psychology
Winning by putting the opponent on your side
Frighten the bureaucrat using your natural endowments
Use logic to prove that obstacles may be bypassed
Law is to forbid illegal acts, and anything not forbidden is legal

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them

Know the rules better than the bureaucrat in charge of regulating your requests

Defeat the opponent using game theory, logic, and psychology

Winning by putting the opponent on your side

Frighten the bureaucrat using your natural endowments

Use logic to prove that obstacles may be bypassed

Law is to forbid illegal acts, and anything not forbidden is legal

The problem with Catch 22: inconsistent laws

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them
Know the rules better than the bureaucrat in charge of regulating your requests
Defeat the opponent using game theory, logic, and psychology
Winning by putting the opponent on your side
Frighten the bureaucrat using your natural endowments
Use logic to prove that obstacles may be bypassed
Law is to forbid illegal acts, and anything not forbidden is legal
The problem with Catch 22: inconsistent laws
Instill a feeling of guilt into bureaucrats by pointing out the inconsistency of rules

The science of creative bureaucracy

Learn the rules, and remove the opaque veil that the bureaucrats carefully maintain between you and them
Know the rules better than the bureaucrat in charge of regulating your requests
Defeat the opponent using game theory, logic, and psychology
Winning by putting the opponent on your side
Frighten the bureaucrat using your natural endowments
Use logic to prove that obstacles may be bypassed
Law is to forbid illegal acts, and anything not forbidden is legal
The problem with Catch 22: inconsistent laws
Instill a feeling of guilt into bureaucrats by pointing out the inconsistency of rules
This is a dangerous path, sometimes it is better to keep your mouth shut

The craft of creative bureaucracy

The craft of creative bureaucracy

Do not fill non-mandatory fields in questionnaires

The craft of creative bureaucracy

Do not fill non-mandatory fields in questionnaires
How Turing won the game of firearm training without enlistment

The craft of creative bureaucracy

Do not fill non-mandatory fields in questionnaires
How Turing won the game of firearm training without enlistment
Use epistemic logic to profit of your opponent ignorance

The craft of creative bureaucracy

Do not fill non-mandatory fields in questionnaires
How Turing won the game of firearm training without enlistment
Use epistemic logic to profit of your opponent ignorance
Don’t lie, but sometimes keep silent with a poker face

The craft of creative bureaucracy

Do not fill non-mandatory fields in questionnaires
How Turing won the game of firearm training without enlistment
Use epistemic logic to profit of your opponent ignorance
Don’t lie, but sometimes keep silent with a poker face
But be persistent, and get your ways by hook or by crook

The game of creative bureaucracy

The art of detecting loopholes and using them to bypass obstacles

The game of creative bureaucracy

The art of detecting loopholes and using them to bypass obstacles
Similar to finding a lemma that interpolates between your capabilities and your requirements. Its proof is a legal course of action, but may be complex with use of exceptions and exceptions of exceptions, etc.

The game of creative bureaucracy

The art of detecting loopholes and using them to bypass obstacles
Similar to finding a lemma that interpolates between your capabilities and your requirements. Its proof is a legal course of action, but may be complex with use of exceptions and exceptions of exceptions, etc. Explain your manoeuvers to the bureaucrat in charge by a proof that every step in them is locally legal

The game of creative bureaucracy

The art of detecting loopholes and using them to bypass obstacles
Similar to finding a lemma that interpolates between your capabilities and your requirements. Its proof is a legal course of action, but may be complex with use of exceptions and exceptions of exceptions, etc. Explain your manoeuvers to the bureaucrat in charge by a proof that every step in them is locally legal
Then have him or her repeat the argument, step by step

The game of creative bureaucracy

The art of detecting loopholes and using them to bypass obstacles
Similar to finding a lemma that interpolates between your capabilities and your requirements. Its proof is a legal course of action, but may be complex with use of exceptions and exceptions of exceptions, etc. Explain your manoeuvers to the bureaucrat in charge by a proof that every step in them is locally legal
Then have him or her repeat the argument, step by step
Nod approval at every step

The game of creative bureaucracy

The art of detecting loopholes and using them to bypass obstacles
Similar to finding a lemma that interpolates between your capabilities and your requirements. Its proof is a legal course of action, but may be complex with use of exceptions and exceptions of exceptions, etc. Explain your manoeuvers to the bureaucrat in charge by a proof that every step in them is locally legal
Then have him or her repeat the argument, step by step
Nod approval at every step
At QED jump and exclaim “you got it”

The game of creative bureaucracy

The art of detecting loopholes and using them to bypass obstacles

Similar to finding a lemma that interpolates between your capabilities and your requirements. Its proof is a legal course of action, but may be complex with use of exceptions and exceptions of exceptions, etc. Explain your manoeuvers to the bureaucrat in charge by a proof that every step in them is locally legal

Then have him or her repeat the argument, step by step

Nod approval at every step

At QED jump and exclaim “you got it”

Now act as if the bureaucrat had the idea in the first place, compliment him/her for it, and state your obedience to the process: Yes, we can!

Mandatory readings

- Joseph Heller. Catch 22
- Robert Pirsig. Zen and the Art of Motorcycle Maintenance
- Eugen Herrigel. Zen and the Art of Archery
- Jerry Rubin. Do it!
- Edward Conze. The Memoirs of a Modern Gnostic