About "trivial" software patents: the IsNot case

Jan Bergstra

Paul Klint

Universiteit van Amsterdam

Universiteit Utrecht

CWI

Further info: www.cwi.nl/~paulk/patents

The IsNot patent application *abstract*

- *A system, method and computer-readable medium support the use of a single operator that allows a comparison of two variables to determine if the two variables point to the same location in memory.*
- Application by lead developers of Microsoft's Visual Basic team

The Isnot Patent application 1 of *24 claims*

- *A system for determining if two operands point to different locations in memory, the system comprising: a compiler for receiving source code and generating executable code from the source code, the source code comprising an expression comprising an operator associated with a first operand and a second operand, the expression evaluating to true when the first operand and the second operand point to different memory locations.*

The IsNot Patent Application *analysis*

- Hey, this is about != in C, Java or C#!
- Or about .NE. in Fortran
- Or about BNE in assembler
- Isn't this prior art?
- Does MS really mean that they invented the inequality operator?

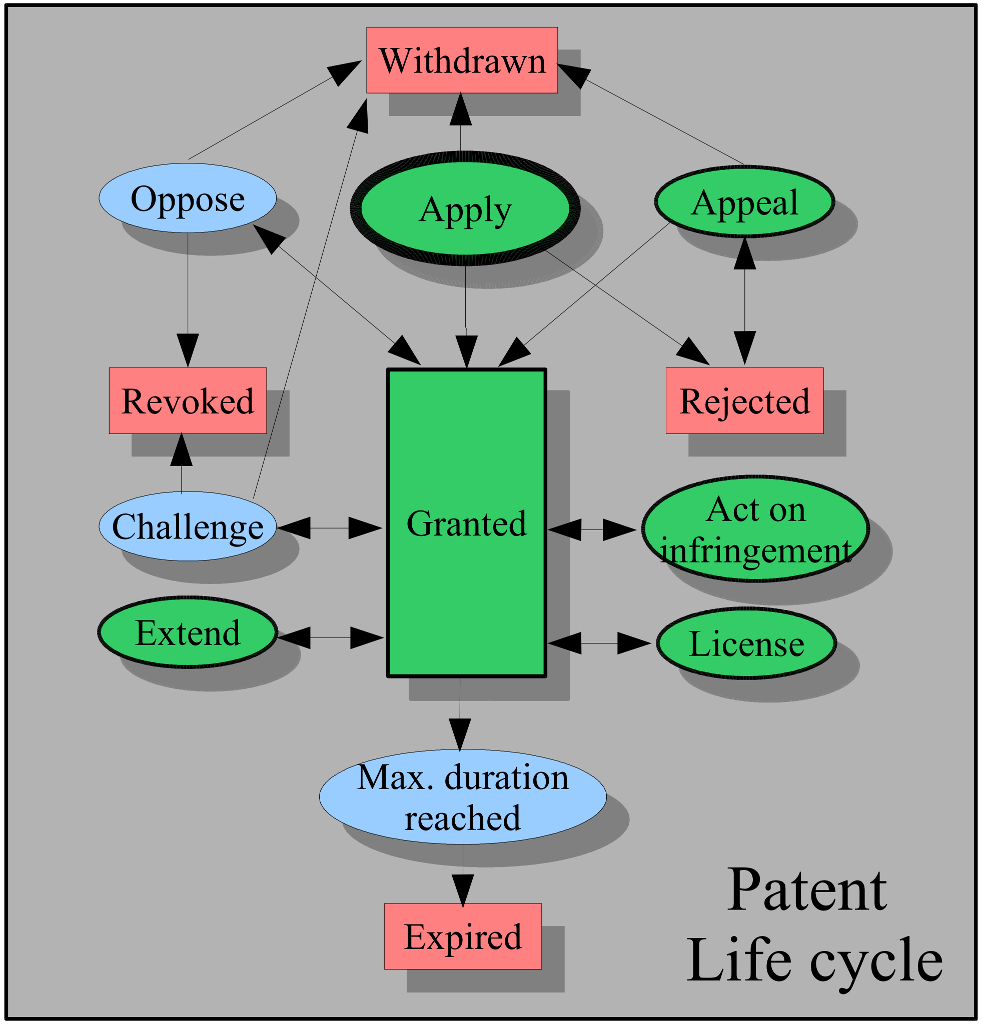
The IsNot Patent Application *analysis*

- Is there some hidden intention in this application?
- Is this about a hidden trick in the Basic compiler?
- Is the intention to challenge the patent system?
  - You must agree: this is a beauty in its simplicity
- We don't know!

We have written an Open Letter to Microsoft to clarify this, see www.cwi.nl/~paulk/patents

How can we reconcile the patent system and the Software Engineering Life Cycle?

Patent Life Cycle

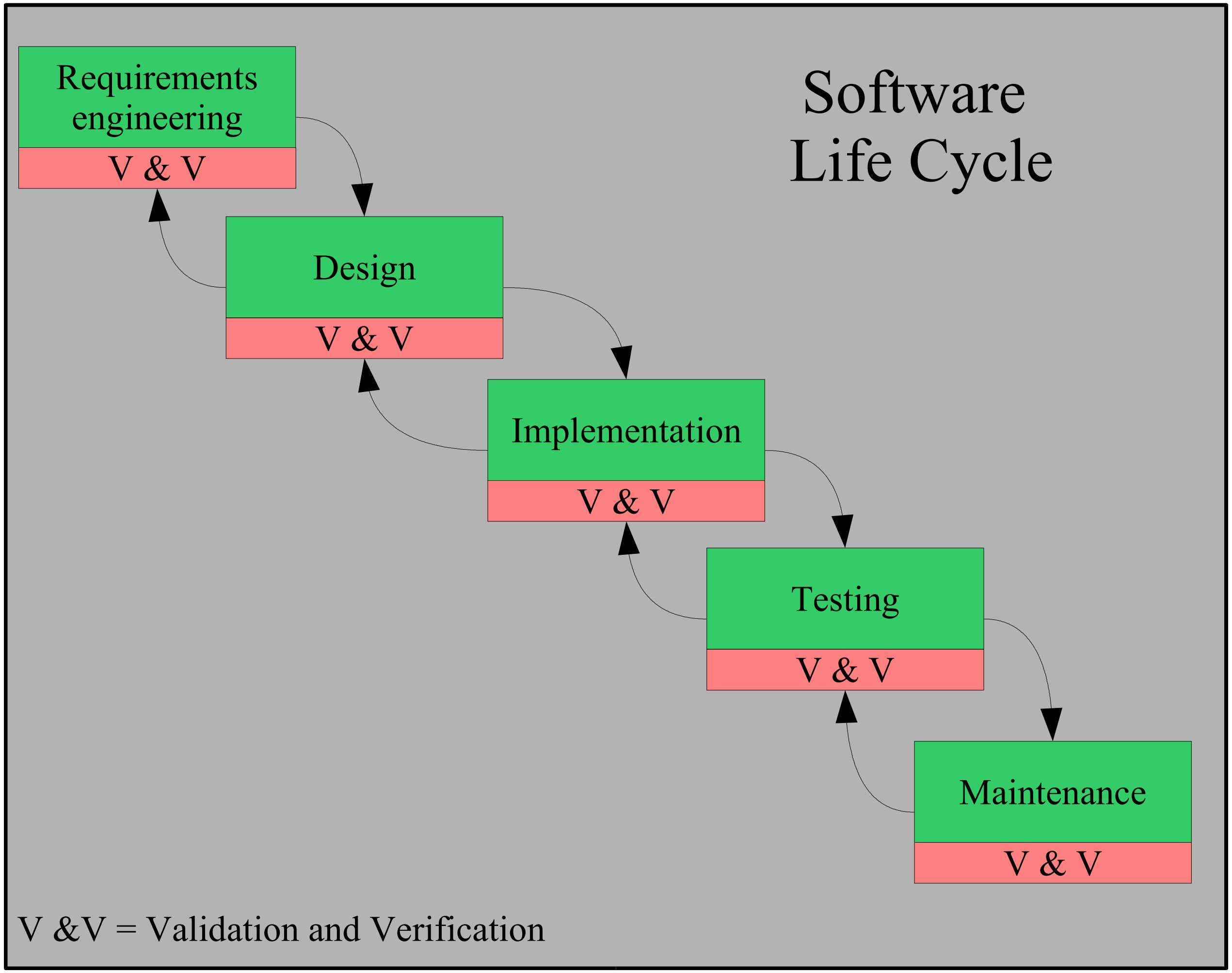


🟢 = action by applicant

🔵 = action by third party

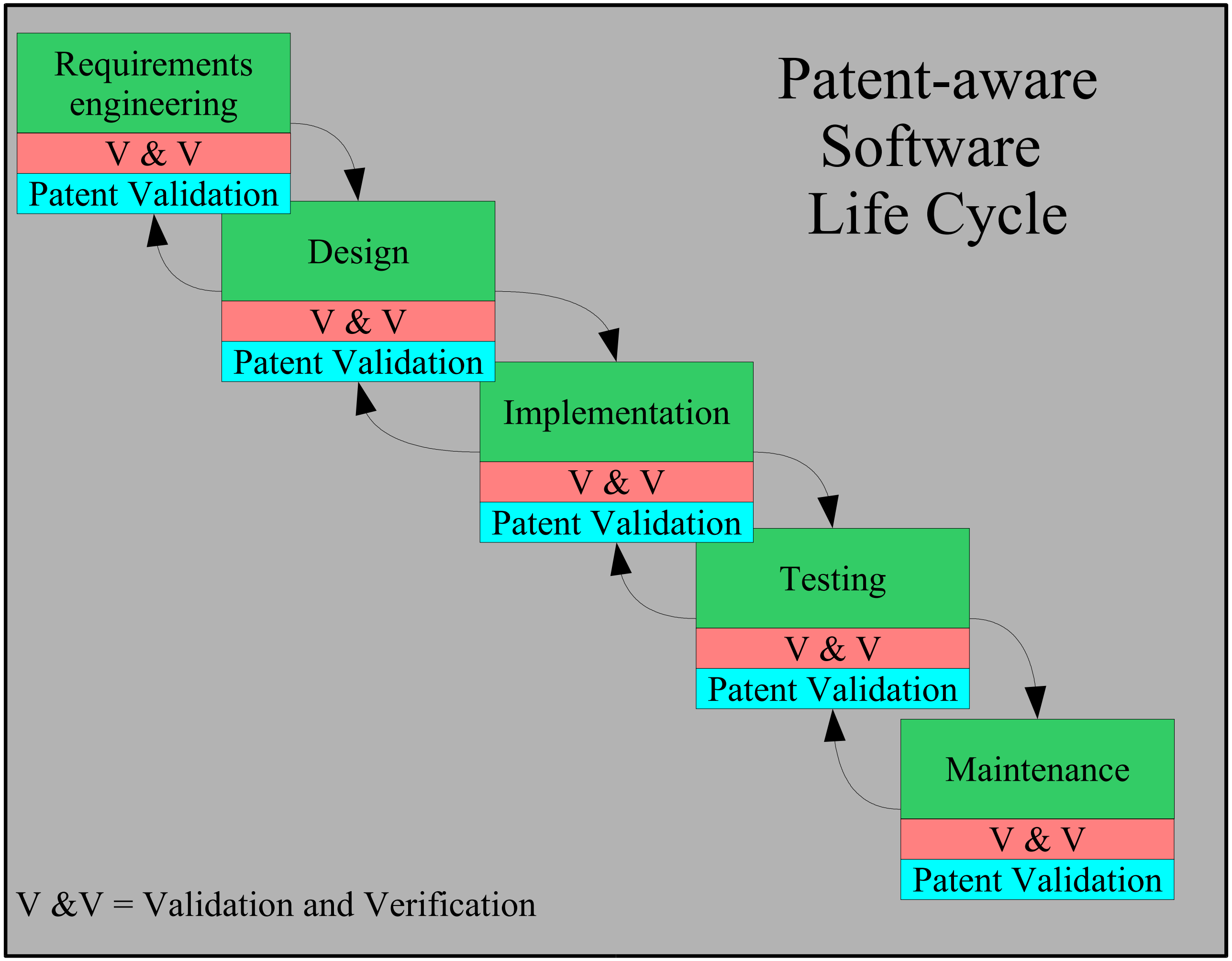
🟥 = end of life

Software Life Cycle



V &V = Validation and Verification

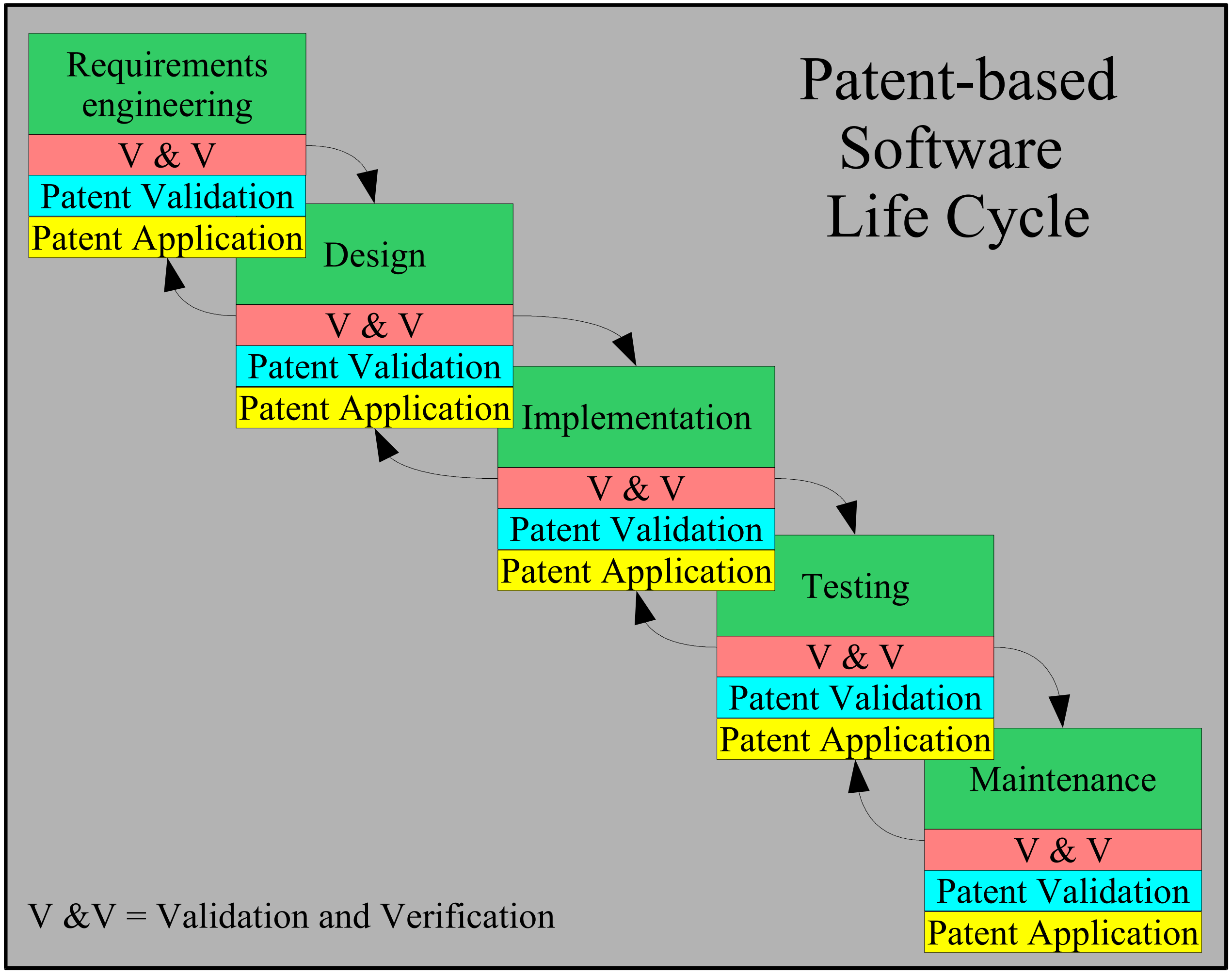
Patent-aware Software Life Cycle *defensive*



Patent-aware Software Life Cycle

V &V = Validation and Verification

Patent-based Software Life Cycle *offensive*



Patent-based Software Life Cycle

Requirements engineering

V & V

Patent Validation

Patent Application

Design

V & V

Patent Validation

Patent Application

Implementation

V & V

Patent Validation

Patent Application

Testing

V & V

Patent Validation

Patent Application

Maintenance

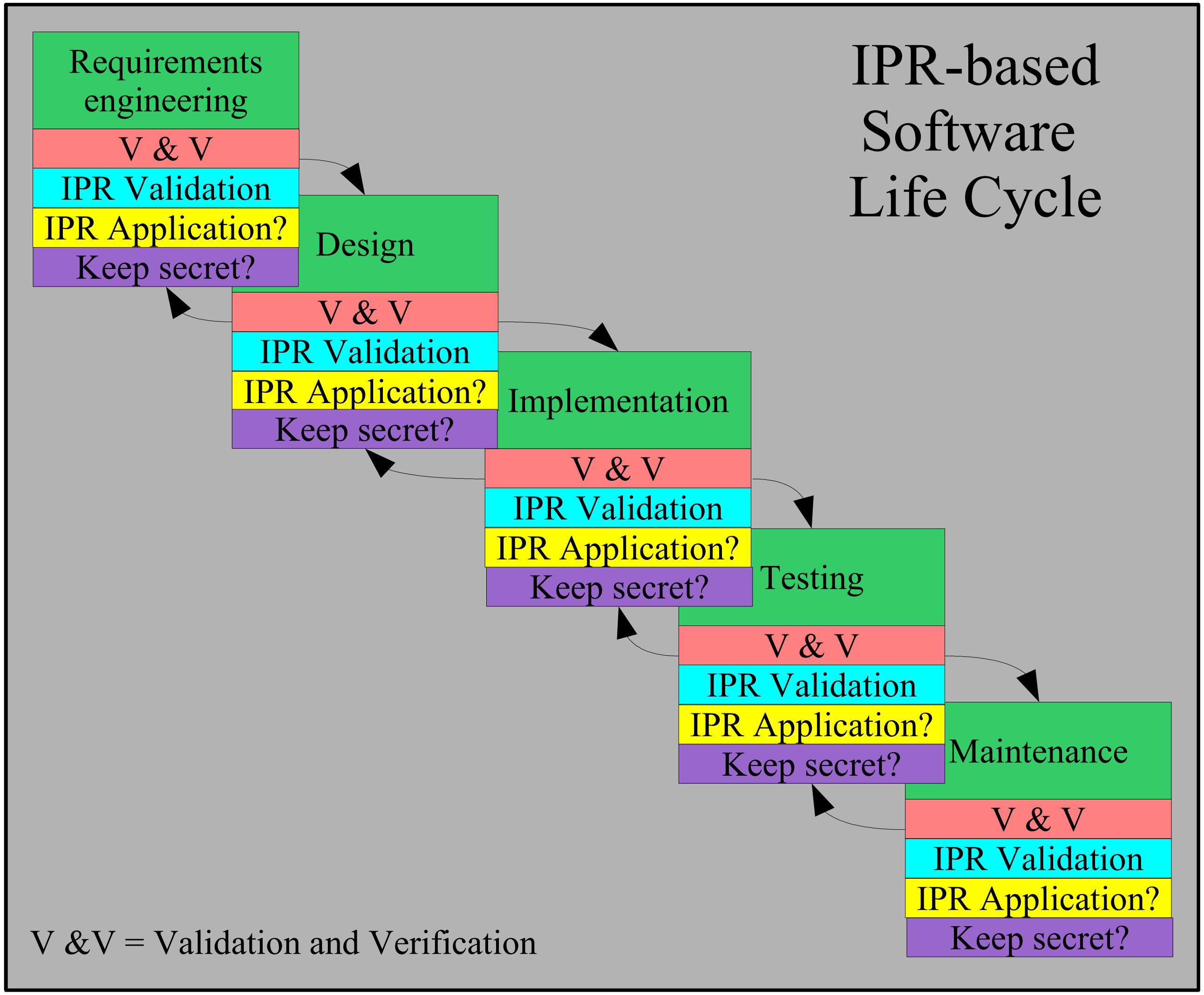
V & V

Patent Validation

Patent Application

V &V = Validation and Verification

IPR-based Software Life Cycle *offensive*



Observations

- Status of prior art and claims is unclear
- Software patenting badly needs input from software engineers and is a topic for research:
  - formalization of prior art and claims
  - inventory of all prior art related to software
  - alternative patenting systems
  - automatic infringement detection
- Publicly analyse and annotate software patents: Gauss project (http://gauss.ffii.org/GaussFrontPage)

Time for Discussion

OSS =

Open source: Sense and Simplicity for the software engineer
(suggested by a Philips researcher)

Further info: www.cwi.nl/~paulk/patents