

On VSTTE: A Reformulation, a Widening and a Focus

Dines Bjørner
School of Information Science
Japan Advanced Institute of Science and Technology
1-1, Asahidai, Tatsunokuchi
Nomi, Ishikawa, Japan 923-1292

March 31, 2006

Abstract

We reformulate the VSTTE (Verified Software: Theories, Tools, Experiments (aka: “Verifying Compiler”)) Grand Challenge. We widen the scope, and we focus it (the project) sharply. At least so we think — for all three points.

We immodestly submit that this reformulation, suitably edited “together” with documents [1,2] mentioned in Sect. 1, could serve as a rallying (a “root”) document for all VSTTE research directions’ documents (aka. “road maps”). Each such more detailed research direction then outline a variety of issues which appears to be relevant for study and resolution by the VSTTE endeavour.

Our immodesty above is rooted in some rather puzzling dissatisfaction with what we perceive as the current directions of the Verifying Compiler (cum VSTTE) initiative.

Contents

1	Background	2
2	The Reformulation, Widening and Focus	2
2.1	The Reformulation	2
2.2	The Widening	2
2.3	The Focus	3
3	Possible Application Domains	3
3.1	Multi-Modal Transportation	3
3.1.1	Domain Theory	3
3.1.2	Sets of Requirements	3
3.2	Other Possible Domains	4
4	Discussion	4
4.1	From Domains via Requirements to Software	4
4.2	On Our Inability to Achieve All	4
4.3	Varieties of Problem Frames	5
4.4	Varieties of Domain and Requirements Specification Languages	5
4.5	Varieties of Programming Paradigms	6

4.6	Integrating Formal Techniques (aka IFMs)	6
4.7	Capturing 80% of the Computing Science Community	6
5	Etcetera	6
A	Other Possible Domains	7
A.1	Financial Service Industry	7
A.1.1	Domain Theory	7
A.1.2	Sets of Requirements	7
A.2	Health Care	7
A.2.1	Domain Theory	7
A.2.2	Sets of Requirements	7

1 Background

We invite the reader to acquaint themselves with the VSTTE effort:

1. Tony Hoare. The verifying compiler: A grand challenge for computing research. J. ACM, 50(1):63–69, 2003.
2. http://research.microsoft.com/~thoare/The_Verifying_Compiler.ppt
3. VSTTE: <http://vstte.ethz.ch/papers.html>

2 The Reformulation, Widening and Focus

2.1 The Reformulation

We propose to reformulate the VSTTE Grand Challenge as follows:

Develop a set of (3–5) software packages for a specific application domain. Each such software package reflects a distinct problem frame. All software packages are programmed in the same programming language. These programs (i.e., the software packages) are annotated — and the language compiler verifies those constructions. The program assertions are either linked to formal specifications of requirements, or themselves express such, typically machine, requirements (as performance, dependability, etc.). The links to formal requirements are verified, but not (necessarily by the verifying compiler). The formal requirements are (otherwise) shown to be a computable model of a subset domain description (which, hence, is also formalised).

2.2 The Widening

Thus, from the above reformulation, we see a widening:

- From a narrower scope, so we understand it and claim, of just the program coding and compilation process,
- to a wider scope

- that “starts” by describing and formalising, to the extent feasible, a chosen domain,
- which then goes on to prescribing and formalising, to the extent feasible, three to five sets of chosen applications,
- and “ends” with the program coding and compilation process.

2.3 The Focus

The focus takes several forms:

- From the outset a specific application domain is identified — one that Ms. and Mr. Taxpayer can relate to.
- From the outset a specific set of three to five distinct problem frame applications (i.e., software packages) are targeted, problems that Mr. and Ms. Taxpayer can relate to.
- So there is a new focus on the VSTTE endeavour studying an appropriate domain and creating a number of 1–2–3–... commensurable (formalised), to the extent feasible, theories of these.
- And there is a new focus on establishing (formalised, wherever feasible) requirements and relating these (formally, wherever feasible) to the domain theories.
- And, finally there is a new focus on relating the formalised requirements to the annotated programmes.

3 Possible Application Domains

To give an idea of the binding of the VSTTE project to a specific application domain we here suggest one such.

3.1 Multi-Modal Transportation

By multi-modal transportation we mean transportation that spans several modes: road, rail, air and sea-transportation.

3.1.1 Domain Theory

A domain theory would build on a domain “model” which captured transportation nets (segments and junctions), vehicles, traffic, possibly timetables, etc.

3.1.2 Sets of Requirements

We can think of the following rather distinct sets of requirements:

- net (road and/or rail line) maintenance,
- net logistics: multi-modal route planning, freight tracking, etc.,
- traffic monitoring and control (rail signalling and station interlocking, road intersection signals, air traffic control, etc.),

- timetable scheduling and rescheduling, and
- net visualisation and traffic animation.

3.2 Other Possible Domains

An appendix suggests other domains.

4 Discussion

Our discussion contrasts the above proposal with our perceived view of the currently emerging “road maps”.

4.1 From Domains via Requirements to Software

There is no denying it: the three phases of software development are “more-or-less” indispensable, yet the current VSTTE seems only to cater for mostly the latter phase.

We have yet to understand the interplay between requirements specs. and program annotations.

And we have yet to understand what can be formally requirements prescribed, what can be formally domain described, let alone the formal relations between requirements prescriptions and domain descriptions.

4.2 On Our Inability to Achieve All

Above it was hinted that perhaps one can indeed establish formal domain theories. Well. The truth is: Only somewhat approximately.

Michael A. Jackson - from a recent e-mail message

A major difficulty of developing a software-intensive system that interacts with a non-formal physical or human domain is that formalisation is always only approximate. If the system is critical it becomes vital to obtain a very good approximation indeed, and this cannot, in general, be achieved by elaborating the description of domain properties. Instead it becomes necessary to structure a set of domain descriptions, with their attendant requirements and subproblem machines, so that they form an assemblage that in total is a good enough approximation. A simple case is a cascading structure in which the highest element in the cascade describes an idealised domain in which everything works 'as it should' and other elements describe a less idealised domain in which various 'faults' invalidate parts of the assumptions on which the highest element is based.

More generally still, a realistic problem is an assemblage of many subproblems, each addressing a partial requirement on the basis of a partial view of the problem world. The composition of these subproblems raises many concerns such as conflict, interference, incommensurability, and so on, which must be addressed if the subproblems and their solutions are to be combined into the desired system. This complexity of composition inevitably leads to a complexity of software structure. Without addressing this aspect of software structure the developer cannot know

whether or not the behaviour of the programmed machine has a good likelihood of ensuring satisfaction of the problem requirements.

One manifestation of this difficulty is that the 'verifying compiler' version of the GC can be seen to depend on the program having an appropriate structure. If the specification is to be embedded in the program the form of pre- and post-conditions and invariants, the program must possess a structure that can accommodate the specification without introducing so many specification variables that the specification itself becomes unintelligible. To take a trivial and grossly simplified example: if a word processor specification includes an invariant on the document that is to be processed, there ought to be a program component corresponding to the whole document. If there are, instead, only components corresponding to operations on the document there will be no place in the program where the document invariant can be written.

Reasonably clear models of the phenomena and concepts shared between the domain and the machine, arising out of both domain descriptions and requirements prescriptions seem a useful way to identify which program components need be established in order to express and verify suitably invariants.

4.3 Varieties of Problem Frames

Also, there no denying it: the various problem solving techniques, phrased, for example, in terms of Michael Jackson's problem frames, cannot be overlooked.

Also the relation between varieties of problem frames within an assumed single domain, as well as the varieties of domains for which requirements and software imply a specific problem frame need be better understood.

Finally, the relationships between MJ's problem frame approach to program construction and the verifying compiler approaches need also be better understood.

4.4 Varieties of Domain and Requirements Specification Languages

Likewise, there is no way one can deny some initial role for several of the current formal specification languages, to wit:

- Yuri Gurevich's original ASM (cf. recent Wolfgang Reisig papers)
- B and event B
- CSP
- CafeOBJ
- CASL
- DC: Duration Calculi
- VDM-SL
- RAISE SL (RSL)
- TLA+
- Z

A VSTTE effort which denies this spectrum is not initially believable.

4.5 Varieties of Programming Paradigms

The regular meetings, for example, of IFIP WG2.3, to me, clearly shows that there are many potentially very relevant approaches to construction of correct programs.

An aim of the spread of three to five application packages is to provide “room” for experiments with these approaches.

4.6 Integrating Formal Techniques (aka IFMs)

And there is no way denying it: We are far from being in a situation where we understand the satisfaction relations needed between various formalisms when specifying domains and requirements: Across the spectrum of the “conventional” specification languages mentioned just above and the graphical formalisms of Petri nets, message and live sequence charts, statecharts, and ER/UML Diagrams.

4.7 Capturing 80% of the Computing Science Community

The VSTTE effort must capture the enthusiastic interests of bona fide researchers in all the countries that we normally think of as contributing to our science — and to do so it must, we believe, widen the scope while focusing, from early on, i.e, from this year, on targeted, widely known applications.

The VSTTE effort is primarily anchored in computing science: the study and knowledge of how to construct software (etc.). But the VSTTE effort must also capture the minds of computer scientists (ie., those who study which “things” can exist “inside” computers).

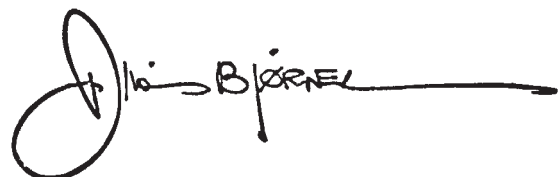
Finally the VSTTE effort must capture the interests of young researchers in emerging countries and their computer science departments.

I would wish to see 100 departments each implementing their versions of the verifying compiler, each deploying their “favourite” specification language(s), and each coding up two or more of the application packages.

I would also wish to see some, for example, a third of these groups and individuals also contributing to the domain theory facets of VSTTE, and some, for example, two thirds, also contributing to the requirements and/or problem frame facets of VSTTE.

5 Etcetera

This draft document will now be circulated. First (21 March 2006) to a very few, named persons, to weed out possible misunderstandings.¹ Then as such are identified and corrected, and if it is still relevant, to a less controlled list of recipients.

A handwritten signature in black ink, appearing to read 'Dines Bjørner', with a long horizontal line extending to the right.

Kanazawa, Japan, March 31, 2006

¹The first such “reviewer” simply suggested to further circulate this document. This is now done.

A Other Possible Domains

A.1 Financial Service Industry

By the financial service industry we understand the individual entities of banks, insurance companies, brokers and traders of securities instruments, stock (etc.) exchanges, portfolio management, etc., as well as the interplay between these, clients, regulatory agencies, etc.

A.1.1 Domain Theory

A domain specification would, as a minimum, cover banks, brokers and traders of securities instruments, and stock (etc.) exchanges, as well as the interplay between these and clients.

A.1.2 Sets of Requirements

We can think of the following rather distinct sets of requirements:

- safe and secure Internet banking,
- more-or-less automated stock brokering and exchange trading (the client view), and
- monitoring the trading of listed stocks (the exchange view).

A.2 Health Care

By health care we understand the individual entities of healthy and sick people (i.e., patients), private physicians, neighbourhood pharmacies, hospitals, recovery and convalescent clinics, etc., and the interplay between these, the regulatory agencies, etc.

A.2.1 Domain Theory

A domain specification would, as a minimum, cover patients, private physicians, pharmacies and hospitals.

A.2.2 Sets of Requirements

We can think of the following rather distinct sets of requirements:

- (some subset of operations upon) patient medical records, i.e., the electronic patient journals,
- hospitalisation plans, and
- scheduling of patient treatment across the spectrum.