

MMM and the SMVS Project

A history of MMM and the SMVS project from 1983-present

Aaron Turner, (last revised) 19 February 2008

Abstract

In June 2006, Christopher Sykes¹, a TV journalist making a documentary for Channel 4 about MMM co-founder Aubrey de Grey² (and specifically about his work as a theoretical biologist³), asked MMM co-founder Aaron Turner for some background about Aubrey's time working at MMM as a practical computer scientist. The present document is derived from Aaron's response to that request, and covers, in narrative form, the 1985 genesis of the SMVS formal verification project at Sinclair Research in Cambridge (UK), the birth of MMM, and the subsequent 20-year+ history of MMM and the SMVS project.

¹ <http://www.sykes.easynet.co.uk>

² http://en.wikipedia.org/wiki/Aubrey_de_Grey

³ <http://video.google.com/videoplay?docid=-3329065877451441972>

MMM and the SMVS Project

SRL

It all started at Sinclair Research Ltd. I joined SRL on 4 July 1983, straight out of university. Firstly, you have to understand that SRL was like no other place on Earth - a total of just 50 or so employees (at that time), mostly young, all eminently capable, generating revenues of about £1m per person and headed by a very famous entrepreneur, of whom we were all extremely proud. Even the building - 25 Willis Rd, Cambridge - looked like a spaceship, with its stainless steel exterior, three-storey atrium and elegant roof garden. The company had lots of money to spend on everything, so we were all completely spoiled. Free cooked meals every lunchtime in the cafeteria. Company parties about once a month. Lots of get-togethers in pubs and eateries in and around Cambridge, usually several times per week, often facilitated by the company's taxi account. But that's not to say we didn't work, quite the opposite - the problem was getting people to go home, or even to stay on holiday. Everybody just loved the company, loved working with such a great group of people in such a creatively permissive and flexible environment. We would work in the evenings, and at weekends. In the computer lab, if you had an idea, however radical, you could just go with it - pushing the envelope and challenging conventional wisdom were considered the norm. And the working environment was, to say the least, laid back. For example, in the middle of my first ever design meeting (for the QL computer, which was under development at the time), the Managing Director, Nigel Searle, folded his arms on the table and went to sleep. Nobody took a blind bit of notice. And I vividly remember one balmy summer evening in the lab, everybody was happily beaver away on their computer screens, while one of the software engineers vamped away on an electronic keyboard and the chief computer designer, David Karlin, was in the corner jamming on bass guitar (periodically attending to his chip design whenever his computer completed some processing task). "So, this is industry", I thought.

Fifth Generation computing

At the epicentre of this wonderful environment was the computer lab (comprising just 9 people when I joined) - the fundamental source of the company's primary income stream. Our focus, of course, was on the company's mainstream computer products - the Spectrum and QL, plus various peripherals, etc. But (Sir) Clive (Sinclair) was very keen to develop a "Fifth Generation" (i.e. Artificially Intelligent) computer in response to the perceived Japanese threat in this field (which still persists, actually). He believed that only massively parallel processing (i.e. lots of CPUs working together) would be able to deliver the enormous processing power that such a computer would require. Thinking several steps ahead, he concluded that Wafer Scale Integration (WSI) would be the only way to implement massively parallel processing while at the same time keeping total system costs low enough to be able to reach a mass market. Normally, several hundred identical ICs (integrated circuits, or "chips"), such as memory devices or CPUs, are manufactured on a single wafer of silicon. Only a proportion of the chips on a wafer will work correctly, due to unavoidable defects introduced during the manufacturing process, for example airborne dust particles messing up the lithography (hence all the "cleanroom" precautions taken in semiconductor fabrication plants). Usually, the individual chips on a wafer are probed by automated test equipment (and the faulty ones marked with a blob of red ink), the wafers are then cut up, and the working chips are enclosed in protective

MMM and the SMVS Project

plastic or ceramic packages (which is the form we finally see them in when soldered onto a printed circuit board). With WSI, the wafers are left intact, and the individual circuits are designed in such a way that the working chips are able to connect themselves together and operate correctly even though some of their neighbours may be defective. The net result is a considerable cost saving during manufacturing because the wafer doesn't need to be chopped up and then re-packaged. Sounds easy (maybe), but it's actually incredibly difficult, not just designing the basic reconfigurable circuits but also distributing power and clock signals across an entire wafer and then getting rid of all the heat. Working WSI had in fact eluded even the largest semiconductor companies for decades, and the prevailing wisdom at the time was that WSI was simply beyond the limits of available technology. But, at SRL, all things were considered possible. Artificial Intelligence, massively parallel processors and Wafer Scale Integration - this is exactly the kind of "pushing the envelope" and "challenging conventional wisdom" that SRL was all about.

MetaLab

As if this wasn't bold enough, Clive decided that this and other blue-sky projects required a proper, world-class research facility. So SRL bought a beautiful old country house - Milton Hall - and started converting it into "MetaLab". As usual, no expense was spared. The grounds were beautifully landscaped (I remember fully grown trees arriving on lorries). Extensions were built, and the joist ends were, of course, gold-plated. There was a new, much larger, computer lab, with offices down one side and a row of workbenches on the main floor. I distinctly remember the chemistry lab. The company's two chemists had different views on how a chemistry bench should be finished, so one was varnished in one way, and one was varnished in another, each perfectly tailored for the individual chemist. There was also talk of a cellar bar and a swimming pool. All usual SRL style.⁴ An entire team - referred to internally as, what else, the Wafer Boys - was acquired from Burroughs in order to realise the WSI dream. The WSI team was headed by controversial electronics engineer Ivor Catt (who was of course immediately nicknamed "I've-a Dog"). Clive naturally realised that his AI computer would need some AI software to run on all his elaborate WSI hardware, and he initially selected me and one other guy (Martin Brennan) from the computer lab to "take care of that minor detail" (although obviously our group was expected to grow into a larger team as the project progressed). So the two of us arrived at Milton Hall when it was still being refurbished, and for a month or so we pretty much had the entire place to ourselves. I remember I selected as my office a high-ceilinged 40' by 30' room with a large bay window at one end overlooking the grounds, and I positioned my desk about three quarters the way inside the room, so that anyone coming to see me had to walk about 30 feet just to get to my desk. Ah the power! Eventually, though, lots of people came over from Willis Rd, and I ended up in one of the offices in the main computer lab. The home computer market had taken something of a dive, and it was all hands to the pump to keep our beautiful company afloat, which meant that I often had to interleave my Fifth Generation work with more conventional, home computer-related projects, particularly - and prophetically - the Spectrum 128.

⁴ not many people know this, but... the C5 electric car was developed by SRL's sister company, Sinclair Vehicles, and 3 prototypes (including a super-fast, never-sold-to-the-public, twin-batteries-wired-in-parallel version) were kept at Milton Hall, whose wide-radius circular drive made an excellent late night racetrack!

MMM and the SMVS Project

AI and formal verification

Starting with only a very basic understanding of AI (I had done the AI course in my third year at university), I spent months trying to work out what "intelligence" was and how a computer might be persuaded to exhibit it. Having read everything I could both get my hands on and understand (a tiny subset of everything I could get my hands on), I distilled it all down to "Aaron's Model of AI" (AMAI), which consisted of four basic processes - Perception, Planning, Reasoning and Learning - all sharing information via a central "knowledge base". Ridiculously crude, yes, but I could "see" in my mind's eye how, if each component were correctly implemented, the overall system could exhibit genuinely intelligent behaviour. (And don't listen to all those hifalutin arguments about "ah but does it really understand, *inside*?" For the purposes of constructing an artefact that does something sufficiently useful that someone would be happy to pay money for it - otherwise known as a *product* - all that matters is what it does when viewed from the *outside*. For all I know, the intelligent human in front of me in the supermarket is actually a perfectly lifelike robot, I can't tell (I'm just assuming it's a human). But whatever it is, however it works inside, from my perspective it's intelligent, because that's how it's *behaving*.) It was pretty obvious that AMAI was so enormous an undertaking it would have to be implemented as a sequence of products. Each interim AI-based product would focus on (some aspect of) one of the four AMAI processes and thereby represent an incremental advance on its predecessor, plus of course, if intelligently planned, each interim product would generate some revenues to keep the project (or even SRL) going. Sounds like a plan, but where to start? Which component of AMAI should we focus on first? The answer was provided for us by another project within SRL, namely the implementation of the Spectrum 128 editor (by a couple of outside contractors known affectionately as the Tazmen). I somehow found myself in charge of managing the QA for this project, and as everyone in the computer lab tested each new version, I was fed the details of each bug so that we might track each one and gradually eliminate them one-by-one. Sounds easy, eh? But each time the Tazmen fixed one bug, they somehow introduced one or two more, and we spent weeks basically going around in circles. It was extremely frustrating for everyone, not least the poor Tazmen, who were incredibly enthusiastic, likable guys. But the whole episode suggested a possible first interim AI product, namely a highly-automated tool to facilitate the rigorous and exhaustive verification of computer software. The fact that such an application would focus on the Reasoning component of AMAI also made good sense, because that particular process (in an AI context) basically boils down to mathematical theorem-proving, and - in stark contrast with the other components of AMAI - mathematics has thousands of years of history behind it and so we would be building on an already well-developed foundation. Even so, I found myself once more contemplating problems of which I knew almost nothing, except for their famed difficulty. As always, I hit the books, and tried desperately over several months to acquire a working understanding of formal logic, mechanical theorem-proving, formal (programming language) semantics, and formal (program) verification. But these are by no means easy subjects to assimilate, and I quickly found myself not quite floundering but nevertheless progressing at a snail's pace.

MMM and the SMVS Project

Enter Aubrey

Around the time that I was planning to move from Willis Rd to Milton Hall, I started hearing about some fabulous new guy, "Aubrey David Nicholas Jasper de Grey" - if that really was his name - who was being hired specifically to join the AI software team, having done some AI thing or other in his final year at Cambridge. Say what? Not only was this guy obviously going to be frighteningly clever, having read Computer Science at Cambridge (as opposed to solid-but-nevertheless-relatively-humdrum Queen Mary College), but he also had genuine expertise in AI, whereas I was essentially a self-taught amateur trying desperately to appear otherwise. It looked as though my brief but highly-enjoyable time as one of Clive's favourite AI chosen ones was drawing swiftly to a close. Even worse, I was also getting animated reports about some "pull up a groove and get fabulous" garden party that this Aubrey person had thrown at Trinity Hall, his Cambridge college. Could it be that he was also cooler than me as well? If I wasn't worried before, I certainly was now! Thankfully, all of these anxieties dissipated in about five seconds when we were finally introduced. Aubrey - Aubs to his friends - who seemed every bit as nervous as I was, was basically an instantly-likeable, super-intelligent, ultra-ultra-laid-back, absolute non-conformist. In other words, he was tailor made for a place like SRL.

The SMVS

Upon his arrival at Milton Hall in June 1985, Aubrey was temporarily assigned to the WSI team where he developed software to manipulate a hardware description language in various (fairly straightforward) ways. Three months later, job done, Aubrey was free to join me on the AI stuff. It took me just two hours or so to explain all my many months of thinking up to that point, including AMAI and the strategy of tackling formal verification as an initial AI application, and Aubrey agreed with me on almost every detail. I was fairly relieved - even though I had had to work virtually everything out from scratch, at least it seemed that there was nothing too wrong with my logic. Nevertheless, the goal of substantially automating formal program verification required a far deeper understanding of logic, proof and mathematics than I possessed, which of course is partly why I'd been struggling so. (The other reason, which I should perhaps make clear here, is that the underlying problems are well known by every computer scientist to be "provably unsolvable", meaning it is literally impossible to write a computer program to solve the general case. In 1985, most computer scientists would automatically conclude that it was therefore impossible to write a computer program to solve these problems in a way that yielded net economic benefit⁵. However, this is not impossible, it is merely - as Aubrey and I would soon experience first hand - extremely difficult.) The Computer Science department at Cambridge - where Aubrey had just finished his degree - was (and still is) internationally famous for its research in these areas, for example Larry Paulson's work

⁵ prevailing academic wisdom within the computer science community would not fundamentally change until the announcement of Tony Hoare's *Verification Grand Challenge* almost 20 years later; in the interim, it became apparent that, until we had a finished product that people could play with, most people we might speak to about the SMVS (including potential customers) would either not understand what we were doing or not believe it was possible, consequently MMM became the "dark matter" of the verification universe, developing the target SMVS product more-or-less in secret without publishing any results (in order to protect our competitive advantage) or any significant interaction with the rest of the computing world

MMM and the SMVS Project

on mechanised theorem-proving and Mike Gordon's work on formal specification and verification. Aubrey actually knew these people, had attended their lectures and done their courses. It was immediately obvious that Aubrey was way ahead of me in these matters, and I often had great difficulty during our discussions understanding much more than the gist of what he was saying. And then one fateful day when we had been alternately discussing matters both theoretical and practical while perusing the many "formal methods"-related books I'd accumulated, Aubrey suddenly snapped the book he'd been studying closed and proclaimed "I know how to do it!" From all the pieces of the formal verification jigsaw puzzle assembled in his head he'd been able to construct a path from problem to solution, which he could "see" in his mind's eye very much as I had with AMAI. The only minor difference between the two situations was that Aubrey instantly understood every minute detail of my ideas whereas when Aubrey now tried to describe the solution he'd just "seen" I basically hadn't got a clue what he was talking about. Not that it mattered, because I was then asked to head up the software team for a successor to the QL code-named Tyche. So as I immersed myself in the details of the 68020 microprocessor and related issues, Aubrey pushed ahead with a prototype implementation of his ideas. Every now and then we would get together to talk briefly about his progress but in truth the only thing that matched Aubrey's enthusiasm was my ignorance of exactly what he was doing. Not to worry, I thought, I would just have to hit the books "a bit more" and gradually catch up. Then, on 7 April 1986, SRL's troubles finally caught up with her and we were all informed that the key assets - basically all finished products and the Sinclair trademark - had been bought by Amstrad and that the company would gradually be winding down. Many of us had seen this day coming and had taken exploratory interviews at places such as Acorn Computers, then in the very early stages of designing what would eventually become the ARM processor, but for the old Willis Roaders like myself nowhere was ever going to match working at SRL. So two days later Aubrey and I agreed to pursue the formal verification tool project on our own behalf, via what would eventually become Man-Made Minions Ltd. Hence the Sinclair Research / Man-Made Minions Verification System (SMVS) project was born.

Man-Made Minions

So, all of a sudden, Aubrey and I were business partners. Just a few minor concerns: we had absolutely no money, we didn't know anything about business, we had no money, we didn't have our own offices or computers, and - did I mention? - we had no money. Not to worry. As soon as we asked "if it would be OK", Clive immediately gave his blessing to our new venture. Obviously, as regards intellectual property, SRL rightfully owned the "portion" of the SMVS project that had been created while Aubrey and I were SRL employees⁶ (as the deal with Amstrad only included finished products, not work-in-progress), and Aubrey and I (as MMM) would own everything produced subsequently. We would eventually have to sort out some kind of formal licence agreement, but in the big picture that was a mere formality that we could attend to in due course, the important thing was that Clive had given the OK and was on our side. Things at SRL were winding down only very slowly and many of us in the computer lab were performing various "tidying up" roles and still had access to the building, so I just happened to be in the lab

⁶ MMM would ultimately acquire outright ownership of this IP in 2001

MMM and the SMVS Project

when a highly fortuitous phone call arrived. Word had gotten out that a job lot of (then very highly regarded) SRL engineers were suddenly available for work, and a call came into the lab from Verran Micro-Maintenance Ltd, who used to do repair work for SRL. They had a design project they wanted to take forward, and were looking for a suitable engineer to take it on. Because the project involved a single-chip microprocessor, and because I was SRL's in-house "single-chip micro guy", the call was passed to me. Next thing you know, I had a reasonably well-paying contract job and MMM had an income. Even better, SRL's WSI project (along with the company's (for those days) very large and fast DEC VAX 11-780 computer, on which Aubrey had been developing his prototype) was spun out into a new venture, Anamartic, which remained in residence at Milton Hall, and Clive arranged a deal whereby Aubrey would do various system management chores, such as computer backups, in return for free use of the VAX, an office and a phone. So, within a very short space of time, MMM had an office, a phone and access to a computer, as well as an income (via my contract at Verran) through which it might support itself.

In business

Initially, I was able to do my Verran contract work from home (actually my girlfriend's house, an idyllic cottage in a tiny Cambridgeshire village), while Aubrey cycled into Milton Hall every day to work on the prototype (Aubrey greatly prefers cycling - the traffic situation in Cambridge being impossible - and has never learned how to drive). Consequently, Aubrey and I were in constant contact, and would frequently meet in some pub or other for pool, beer (for Aubrey), coke (for me) and talk of all things AI. After several weeks working together, technically, as a partnership, we bought an off-the-shelf limited company. The phone conversation during which we chose the company name will be imprinted on my mind forever. I was reading from a thesaurus a list of words related to "artificial" and "intelligent", and when I got to "man-made" Aubrey suddenly interjected "Oh, what's that word? Begins with 'm', meaning worker'? ... Minion!" And that was it - Man-Made Minions Limited, purveyors of "artificial workers" was born! Of course, now that we owned our own company and were developing our own products, we were no longer supported by an infrastructure comprising lots of nice people who took care of - and insulated us from - all that tedious real-world stuff. Suddenly we were directors and shareholders, we had to work out what these things meant, and we had to appoint auditors and lawyers and file things at Companies House and do our VAT and PAYE returns. Plus we appointed city trademark and patent attorneys Kilburn & Strode to help us negotiate and draft the licensing agreement with SRL. These things are not cheap, and suddenly my contract income was not looking quite so impressive, and we quickly realised that being in business for yourself is not easy - it's all struggle and risk.

Formal verification 101

The immense scale of the task that Aubrey and I had undertaken was also becoming abundantly clear to us. Basically, Aubrey's prototype had to take as input a text string of the general form $\{P\} C_1, C_2, C_3, \dots, C_n \{Q\}$ where P and Q are statements of first-order logic describing conditions on program state (such as " $x = 0$ and $y \leq z$ " meaning "at this point in the program's flow of control, program variable x has value zero and the value of program variable y is less than or equal to the value of program variable z ") and

MMM and the SMVS Project

C_1 to C_n are program "commands" (typically assignment statements, IF...THEN...ELSE conditionals, WHILE loops and procedure calls). What $\{P\} C_1, C_2, C_3, \dots, C_n \{Q\}$ actually asserts is "if the sequence of commands $C_1, C_2, C_3, \dots, C_n$ is started in a state satisfying P then $C_1, C_2, C_3, \dots, C_n$ is guaranteed to terminate in a state satisfying Q ". This can be viewed as a putative mathematical theorem (or statement) requiring formal proof. A computer scientist would refer to this as a statement of total (rather than merely partial) program correctness, because it encompasses the issue of "termination" - i.e. does the program always finish executing in a finite time, or can it get "stuck" in an infinite loop? Obviously, real-world programmers are necessarily concerned with total correctness. The way to (formally) prove total correctness is to effectively execute the program (a) *backwards* and (b) *symbolically*. In other words, Aubrey's prototype first has to propagate Q backwards past C_n thereby generating some Q' such that $\{Q'\} C_n \{Q\}$ is true (and what's more Q has to be as logically "weak" as possible - just using FALSE won't do!) One minor detail: if C_n is or contains a program "loop" statement, then (a) backwards propagation past C_n requires the discovery of a suitable "loop invariant", and (b) in order to prove that C_n always terminates we also need to discover a suitable "loop variant" (and I won't even try to start to get into what these things mean). This backwards propagation is then continued all the way to the beginning of the program, at which point we're left with the proven theorem $\{Q'\} C_1, C_2, C_3, \dots, C_n \{Q\}$. If we can generate Q' in this way and then formally prove that P implies Q' (i.e. whenever P is satisfied, Q' is also satisfied) then we have effectively proved $\{P\} C_1, C_2, C_3, \dots, C_n \{Q\}$, in other words we have formally proved that the program $C_1, C_2, C_3, \dots, C_n$ always satisfies its (formal) specification comprising "pre-condition" P and "post-condition" Q . If we can do all of these things [(1) basic backwards propagation, (2) first-order theorem-proving, (3) loop variant discovery and (4) loop invariant discovery⁷] in a high-enough proportion of cases and using only reasonable amounts of computer memory and CPU time such that end-users experience net economic benefit then we've got a world-altering product - otherwise, we've got nothing! The advantage of formal program proof compared with conventional selective testing? Conventional testing only checks a program against a tiny proportion of all possible starting states (meaning that lots of bugs can be and usually are missed), whereas formal proof checks a program against *all* possible states (meaning, if such a proof can be constructed, that no bugs have been missed and therefore the program is completely bug-free). Putting this into the context of the Tazmen and the Spectrum 128 editor, (a) any bugs in the editor would prevent a proof of correctness from being found, (b) if a proof could be found, it would mean that the editor was bug-free, and (c) if any subsequent refinements to the editor introduced any new bugs then they would immediately be detected by the program-proving process. But only, of course, if the program-proving process could be sufficiently automated to be industrially practicable.

You have to be slightly crazy

Little wonder then that, until very recently, most computer scientists have regarded this problem as "essentially impossible". In theory, technological advances are made as the global collection of engineers and scientists explore every possible path from present to

⁷ where (2)-(4) are all provably unsolvable in the sense described earlier

MMM and the SMVS Project

future technologies, with the fact that the vast majority of these paths are "dead ends" being compensated for by the relative few that ultimately bear fruit. In principle, it's a process of exhaustive search. In practice, not all possibilities are followed. PhD students only have 3 or 4 years to complete their research, and they want to be awarded their degrees at the end of it, so the paths they follow tend to be the "safe" ones. Professional academics rely heavily on small parcels of grant funding, are under continual pressure to publish at frequent intervals and for their publications to be cited by others, and take very seriously their responsibility for extending the frontier of human knowledge with infinite care, consequently they too tend to follow extremely conservative paths of enquiry. And finally, corporate executives in charge of industrial research and development teams are under relentless time pressure to produce results that can be exploited commercially and few are willing to risk their careers by sponsoring overly adventurous projects. So, if the prevailing wisdom is for example that such and such a problem is provably impossible then no-one is likely to go anywhere near it, and - as the history of groundbreaking innovation clearly shows - anyone who does risks being branded a dangerous heretic by their peers. Nevertheless, Aubrey had seen, in his "eureka" moment, feasible algorithmic methods of constructing - in many if not most "real-world" cases - both loop variants and loop invariants. Of the provably unsolvable problems for which we require workable solutions, that only leaves first-order theorem-proving. But, in practice, the vast majority of real-world programs are not deeply mathematical, which means that the theorems involved when trying to prove their correctness may be very large (often covering several pages each!), but they are rarely (mathematically) very "deep". Add to this the fact that computer power per dollar doubles every 18-24 months (an empirical observation known as Moore's Law) and it becomes pretty clear that mechanical theorem-proving (for the purposes of formal verification at least) is not quite the obstacle it at first seems to be. Consequently, it was all doable - all we needed was the stamina to actually do it.

Like climbing Everest with no oxygen and a fridge strapped to your back

In addition to having a genius like Aubrey on your team, this turned out to be the second most important thing required by the project - an infinite amount of stamina. When you are extending the state-of-the-art, Hofstadter's Law always applies: *It always takes longer than you expect, even when you take into account Hofstadter's Law.* So I slogged away at my contract job down in Surrey, and Aubrey slogged away at the prototype up in Cambridge, and every now and then we would meet up for pool, beer and coca-cola, and discuss how our respective things were going. Slowly the weeks turned into months, and the months into years, and still we slogged away. Eventually, Aubrey had implemented sufficient infrastructure to test the gradually evolving prototype against various program routines, to see if it would discover the bugs that he had planted. And then, in August 1988, a milestone - the prototype found its first *real* bug, i.e. a genuine mistake that Aubrey had made as opposed to a "test" bug that he had deliberately inserted. It might have been just a simple bug in a simple program, but nevertheless the bug was detected completely automatically, with all loop variants and invariants discovered automatically and all first-order theorems proved automatically - no one else in the world could do this. But still we knew that in the end we would only have a truly viable business, and only make a genuine, noticeable difference to the computer industry, if (in classic SRL style) we could reach a mass market - at least hundreds of thousands of end-users. And we

MMM and the SMVS Project

knew that this would require a level of performance that we had still not yet achieved, and so we persevered. My technical knowledge was still lagging way behind Aubrey's, so I still didn't 100% understand how the SMVS prototype worked, but nevertheless I was completely happy to continue supporting the project via my contract job. My reasoning was simple: I knew that Aubrey was remarkably intelligent (and highly knowledgeable in this field), I knew that he was firmly focused in a particular - commercially motivated - direction, and I knew that he worked extremely hard every day. Putting all of this together, each day that he spent sitting in front of that VAX terminal he was adding value to a technological asset that would one day have realisable value. Good enough for me. Every time I visited Aubrey, which for purely logistical reasons was becoming less and less frequent, his office was buried in scraps of paper with indecipherable mathematical theorems scribbled on them, each one some kind of trick to squeeze one iota more performance out of the prototype, and many, many such tricks were required. In 1989, after a mere three years of negotiation, we finally signed the formal licence agreement with SRL. It had cost us a fortune, but at last it was done. In the same year, the senior management at Anamartic could no longer satisfactorily explain to their masters at Fujitsu (who had pumped £2m into the company) the continued existence of the apparent squatters sucking vast numbers of CPU cycles out of their VAX. And so we were finally evicted from Milton Hall. MMM then purchased its own computers - identical IBM PS/2s running OS/2 - and rented a house in Tamarin Gardens, Cherry Hinton. This gave Aubrey an office to use during the day and me somewhere to stay when I came up to Cambridge. Each year the long slog continued, and each year the prototype improved. By 1992, the prototype's performance was such that it could now verify, completely automatically, an algorithm known as "bubble sort" (as well as other algorithms of similar complexity) with all loop variants and invariants discovered automatically and all proof obligations (first-order theorems) discharged automatically. Few algorithms that industrial programmers actually write are as logically complex as "bubble sort" (which is why it's a classic computer science textbook example), consequently we judged this milestone as the "coming of age" of the underlying SMVS algorithms. In other words, we had finally achieved the level of automated verification performance for which we were aiming!

Now do it all again, only this time properly

There is a long-standing rule in Software Engineering which goes "always plan to throw one away". It was beginning to dawn on us why. By this time, the SMVS prototype (essentially a research tool for Aubrey's eyes only) had been modified continuously for almost 7 years without any formal quality assurance mechanisms in place. Consequently, although we firmly believed the general-purpose verification algorithms at the heart of the prototype to be sound, its implementation was by now (ironically) riddled with bugs and highly idiosyncratic to use. Aubrey and I took a week off for an MMM "neatification" seminar (a nod to the famous conflict within AI between the "scruffies" and the "neats"). This seminar just happened to be held in the South of France, our only ever MMM jolly. We discussed the possibility of cleaning up the prototype into a world-class product, but came to the reluctant conclusion that we had no choice but to re-write the commercial version entirely from scratch, based on the key algorithms underlying the prototype. Plus there were additional factors. The level of performance exhibited by the prototype had not only confirmed to us that we had achieved a substantive advance on the

MMM and the SMVS Project

state-of-the-art that would subsequently translate into considerable customer benefit, it also broadly suggested how a commercial product might ultimately look and feel. Armed with this information, and in order to measure how the market would react to such a conceptually new product, we conducted a market research survey targeted at over 1000 software development professionals, as well as face-to-face interviews with about a dozen potential customers. We also commissioned, with financial support from the DTI, a study by a professional firm of marketing consultants. The overall results were quite surprising. Most market surveys can expect a response rate of 1-2%, whereas ours achieved over 20%, strongly indicating that we had "touched a nerve" within the software development psyche. On analysing all the data, it became abundantly clear that (a) "push button" automation - as evidenced by the SMVS prototype - was absolutely key to achieving widespread commercial take-up, (b) the market would fiercely resist any requirement to change working practices in any significant way (such as being forced to write formal specifications in advance of coding), and (c) the market would also fiercely resist being forced to use specific or proprietary notations - instead customers insisted on being able to use whatever programming languages they chose. The net result of these market studies was a set of market requirements R1-R9. It was plainly obvious that, if we simplistically implemented our commercial product as a straight, but high-quality, copy of the SMVS prototype, without supporting at least in principle all of R1-R9, sales - and the industrial impact of our verification technology - would ultimately peak at a far lower level than we required in order to achieve our goals⁸. And what is the point of going through hell if you don't achieve your goals at the end of it? So, in addition to having to re-write everything from scratch at the source-code level, it looked like we were also going to have to thoroughly review everything at the higher "architectural" level as well.

Little help?

Some solace was provided by our belief (now that the performance exhibited by the prototype had confirmed that the really difficult problems were all adequately solvable) that we should now, without *too* much effort, be able to secure industrial partners, sponsors and other forms of support to assist us with the (mostly) "mere engineering" work remaining. It was not particularly encouraging, therefore, when a Cambridge DTI official chose not to send us an application form for the potentially £150k SMART award on the basis that we were only a 2-man company and he didn't want to "waste" any forms as he didn't have many left. The IS Director at ICI, without even hearing what our project was about, told me during a telephone conversation that ICI was a world-class company and that there was absolutely no way a 2-man company could have anything they didn't have already. The man in charge of Computer-Aided Software Engineering (CASE) at HP UK informed me during a brief conversation at a computer exhibition that Hewlett-Packard simply could not do business with a 2-man company (ironically, HP is famous for having been started in 1939 by two guys in a garage - Bill Hewlett and Dave Packard). BT's Martlesham Heath research labs were willing to set up a trial application - once we had a fully working product. (And yes they did understand that we were not yet at that stage, it was just an easier way of saying that they didn't want to do business with -

⁸ in marketing theory, this is known as *crossing the chasm* from *early adopters* to the *early majority*, the latter being a much larger and more demanding market segment and therefore much harder to penetrate

MMM and the SMVS Project

you guessed it - a 2-man company.) Disappointed, frustrated, exasperated and increasingly desperate, we organised our own seminar at Peterhouse in Cambridge, which was attended by delegates from a fairly impressive array of organisations, including AEA Technology, BAe Military Aircraft, Hawker-Siddeley Dynamics, Lloyds Register, Siemens-Plessey Defence Systems (SPDS) and SRI International. (We charged each delegate £125, so that we could afford to take everyone to lunch at Brown's!) Everyone at the seminar was extremely polite and attentive, however the consensus of opinion (with one exception) was that the SMVS project was very interesting but (from their perspective) we didn't actually *have* anything yet - in other words, there's not much point in talking to (most) potential customers until you've got an actual, finished, working product for them to play with. The one exception was SPDS, who embarked on an enthusiastic evaluation of the SMVS, including sending someone to Cambridge to see the prototype. It was all looking very promising, until, as a consequence of the collapse of the Berlin Wall and then the U.S.S.R, SPDS was forced to lay off large numbers of people, thus making it near-impossible for any manager to justify supporting external projects. So, not only were we going to have to review the SMVS at the architectural level and re-write everything at the source-code level, we were going to have to do it all on our own.

Role reversal

For the previous seven years, through my IT contract and consultancy work (mainly at Verran, but occasionally elsewhere), I had been bringing fairly reliable revenues into MMM which, combined with frequent contributions from family and friends, was more-or-less sufficient for the company to meet its various ongoing expenses, including modest salaries for Aubrey and myself. With all the really hard research problems basically solved, the nature of the technical issues now facing us was such that I was rather more suited to tackling them than Aubrey. And so we swapped. Aubrey took a job at Hermann Hauser's Active Book Company (part of AT&T's EO Computers since the previous year), contributing a portion of his salary each month to MMM, and I set about looking at the architectural-level problems in the light of all that we had learned so far. Ten months later, through his partner (Adelaide, a geneticist), Aubrey landed a position at Cambridge University's Department of Genetics, providing IT support to the FlyBase project (a database of genetic and molecular data for *Drosophila melanogaster*). It was while immersed in this environment that Aubrey developed his acute interest in genetics, microbiology and biomedical gerontology, ultimately earning his Cambridge PhD.

Abyss

Although the first seven years had been, particularly at times, very difficult and stressful, the next nine would be infinitely worse. As if the search for a substantive solution to arguably the most fundamental and technically difficult problem in computer science (the exhaustive removal of bugs from software source code via automated formal verification) wasn't difficult enough, doing it under conditions of increasingly intense financial anxiety, frustration, disappointment and personal sacrifice have at times made the journey almost unbearable. I once went two months without a wink of sleep. I spent three years sleeping in my car two nights a week, and many more years (my "petrol station food" period) not being able to afford to shop in supermarkets. One summer I had just £1 per

MMM and the SMVS Project

day to spend on food; as a result I now have tremendous sympathy for anyone in the world who is forced by circumstance to endure hunger without having the means, as the vast majority of people in the developed world do, to simply go out and buy something to relieve the pain, and instead have no choice but to simply *be* hungry. At one point, I even made detailed (and very clever) plans for suicide, and I later developed a minor heart condition most probably accelerated by stress. But, as with Vietnam veterans, only a fellow "entrepreneur" who has actually been through it could ever hope to understand. Fundamentally, I was only able to persevere through it all because of the extent to which I believe in the SMVS project. As an experienced software developer, I *know* that virtually every program out there is riddled with bugs that software testing etc was unable to detect prior to deployment. I *know* (from our business planning) that this problem currently costs the world at least \$1 billion every day. And I *know*, having seen Aubrey's prototype perform complex formal verifications completely automatically, that (assuming reliable project funding) MMM possesses the technical basis of a *revolutionary* solution.

Oxford

With Aubrey's continued financial, and occasionally technical, support, I spent nine long years designing an overall system architecture that would accommodate, in principle, all of market requirements R1-R9. During this period, I finally accumulated sufficient knowledge and insight to broadly understand how Aubrey's prototype worked. I spent '95/96 at Oxford University working towards an MSc in Computation. In the end, I wasn't awarded my degree, but it didn't matter. As Aubrey had done while at Cambridge, I had been able to meet and talk with and attend lectures given by some of the greatest minds in computer science, including Tony Hoare (my advisor), Mike Spivey (my supervisor), Joe Stoy and Bill Roscoe. During one lecture given by Joe Stoy, a particular concept of formal proof that I had been struggling with for many years (just trying to learn from books) was suddenly illuminated, and in that one instant my entire year at Oxford was justified. When I later described to Mike Spivey the level of performance our prototype had achieved he looked straight at me and said "I don't believe you". Fair enough, I thought, because we didn't yet have a fully working system with which to prove such a claim, but for me the episode illustrated just how far in advance of the state-of-the-art our project had progressed. For my MSc thesis I was able to explore various problems related to, of course, our formal verification project (rather than one of the "safe" projects offered by the Department), and this was ultimately the main reason I wasn't awarded my degree, which was a shame, but in the big picture the whole reason I was at Oxford in the first place was to progress the SMVS project in every way I could, so it didn't really matter.

Angels

During my time at Oxford, we secured, in collaboration with Vienna-based software quality assurance consultancy APAC GesmbH, a small amount of grant funding through the EC's Esprit programme. Having established an excellent relationship with our Esprit project officer, I was then offered a number of consultancy assignments as a formal "reviewer" of various pan-European Esprit projects, each involving a consortium of large corporations and renowned universities trying to put together some improved formal verification tool (typically by trying to bolt two or more existing systems together in an

MMM and the SMVS Project

attempt to achieve something "greater than the sum of its parts" - a seemingly attractive plan which is *always* a mistake, because, of course, the parts never fit together properly!) None of these consortia seemed to properly understand market requirements R1-R9, if at all, and so I watched on as each of these multi-national, multi-million-euro projects predictably failed to deliver results having any significant commercial potential. Each such experience highlighted even further the correctness of the approach we were taking with the SMVS, and thus added greatly to our confidence. Of course, there were setbacks, as usual. At one point, having paid a substantial amount of money that we couldn't really afford in order to attend an Esprit-related seminar in Brussels designed to put high-tech EU companies in touch with EU Venture Capitalists, the organisers were reluctant to display a poster describing MMM's project because - as you can probably guess - we were only a 2-man company. Eventually, having spent literally 18 months researching and writing a mammoth business plan for the SMVS project, followed a couple of years later by a week-long "Train-IT" business planning course in Germany, followed by a year-long but ultimately abortive period of negotiation with a prospective Business Angel, we finally secured Angel funding from a small group of UK businessmen (the deal being signed, particularly memorably, on the evening of 11 September, 2001)⁹. Interestingly, as part of their due-diligence, our new investors had sought an opinion of our technology from a trusted third-party, who, being a computer scientist, confidently assured them that the problem we were claiming to have solved was well-known to be provably impossible and that consequently he believed that our search for investment was most likely a scam and that we must be extremely sophisticated high-tech grifters! Luckily, with the help of another trusted techie, we were able to get past that particular obstacle, but once again we had been reminded of a recurring truth: until we've actually got a working product to show, we will occasionally, perhaps even frequently, be misunderstood, misjudged and mistrusted, even by our peers. Comes with the territory.

Coding

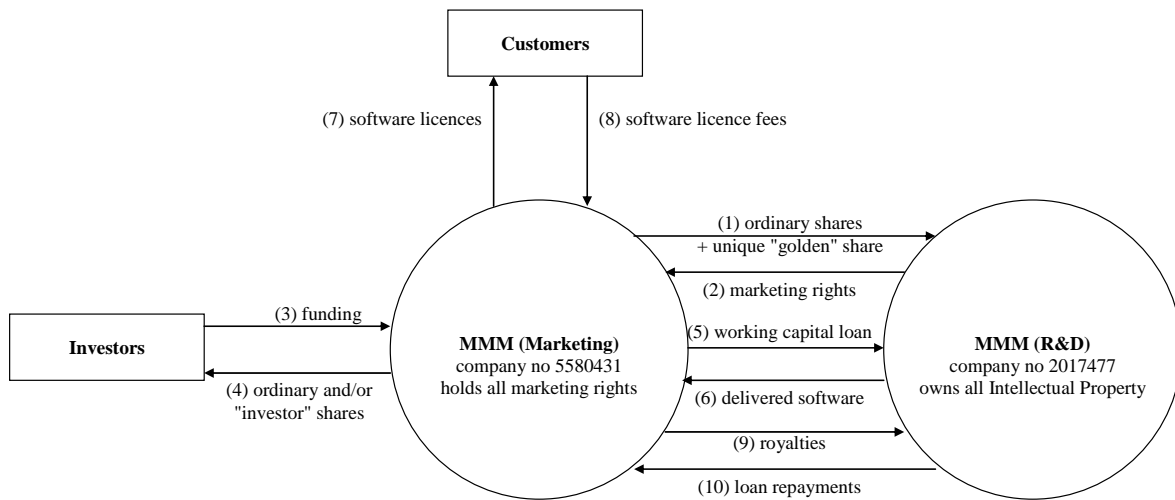
Finally, Angel-assisted, source-level coding of our target, R1-R9-compliant, SMVS product had commenced. Eager to avoid the quality problems that had ultimately plagued our prototype, we established a strict quality management regime. This of course kept our code, as it emerged, nice and clean, but also had the side-effect of significantly slowing the rate at which code was being produced. Naturally, with a system as complex as an automated program verifier, we simply could not cut any corners or take any shortcuts - if we did, the resulting compromises (somehow substandard components or awkward, ill-fitting interfaces) would inevitably feed through to overall system performance. So instead we just carefully laid the foundations piece-by-piece, and gradually the list of SMVS components on my 6' whiteboard turned from green (meaning "not implemented") to red (meaning "implemented"). With only a finite number of components to go, all we had to do was keep slogging away, and they would each turn red one by one. After 20+ years of hard work, risk and sacrifice, it looked like we were finally in the "end-game"!

⁹ in the process, MMM acquired outright ownership of the portion of SMVS-related IP originally vested in SRL - this was necessary so that we could warrant, in the formal contract with the investors, that MMM "owns and shall continue to own all Intellectual Property Rights in connection with and in all Software"

MMM and the SMVS Project

Nothing's ever easy

For around 3½ years our small group of Business Angels kept us going with monthly "drip-fed" funding, meaning that we had absolutely no slack in our finances at all. Even so, one by one the pieces of the SMVS jigsaw puzzle started to fall into place, and, compared with our pre-Angel days, it was like some beautiful dream. But of course even high net worth individuals have cashflow considerations, and in January 2005 (with about 30,000 lines of C source code written and tested so far) the dream came to a rather abrupt end when our Angels were no longer able to continue. With no financial "buffer" and with our accumulated work-in-progress not yet at a point where it actually "did" anything, project progress came to a grinding halt and we were destitute again. Amidst the financial and personal turmoil in which we once more found ourselves, we had to work out some way in which the contribution of our Business Angel investors up to that point was fairly represented and which (very importantly) was also compatible with our longer-term AMAI-related goals. In late 2005, after lengthy negotiations, we finally settled on a two-company scheme whereby MMM (R&D) would (a) hold all the Intellectual Property in the SMVS and (b) effectively grant worldwide marketing rights¹⁰ to MMM (Marketing) of which (at that point) our investors would hold a minor share. This also established a mechanism whereby future investment would enter via Marketing:



This scheme allows us to "share the wealth" with our investors while keeping the IP safe from acquisition and available to use as the foundation for further AMAI-based products.

And then there were three...

In June 2006, after about 15 months of trying to progress the SMVS project while at the same time funding that progress from personal sources, we decided (considering the near impossibility of making any significant progress under such difficult conditions) that we should attempt to raise further external Business Angel (and/or other equity) funding in order that the project might once more progress at a more optimal rate. Progress was slow

¹⁰ for all of R&D's future software products, including any AMAI-based products built on top of the SMVS

MMM and the SMVS Project

at first, but then several successes arrived all in the same month, December 2006, and suddenly the world was looking positively rosy again. We then met a very nice former investment banker and serial entrepreneur, Tim Leeder, who agreed to come on board as MMM (Marketing)'s new Commercial Director, with further (and very substantive) fundraising as his primary initial responsibility. At the time of writing, Tim is using his extensive contacts and business experience to secure progressively larger tranches of project funding, and these efforts are starting to meet with some success. I, meanwhile, am busy progressing the technical implementation of the SMVS product (in anticipation that, as funds permit, "Team MMM" will at some point be augmented with additional development engineers), and Aubrey provides frequent moral and occasionally technical support. As ever, we just keep slogging away. Together, we make the perfect team!

MMM and the SMVS Project

APPENDIX - Program verification timeline

1837	Charles Babbage begins designing his (steam-powered, 30m × 10m!) <i>Analytical Engine</i> (the first ever design for a general-purpose programmable computer), and later considers "verification of the formulae placed on the [operation] cards".
1936	In his paper on the <i>Entscheidungsproblem</i> , Alan Turing introduces the idea of a <i>Turing Machine</i> which he uses to prove the undecidability of first-order logic. This is the first of a series of so-called "negative" results formally proving that the set of all possible programs does <i>not</i> include a program V that is able to decide the formal correctness of (i.e. formally verify) any given program P - any attempt to write a "program verifier" V (returning a yes/no result) is therefore futile because the problem is <i>provably impossible</i> . However, it <i>is</i> still possible to write a program V' (returning a yes/no/maybe result) that is able to formally verify <i>some</i> programs but not <i>all</i> programs, even if it is not possible to define in advance exactly for which programs V' will be able to return "yes/no" rather than "maybe".
1947	John von Neumann co-authors a paper describing how <i>assertions</i> may be used as part of an argument (i.e. proof) that a program produces a specific desired effect.
1949	Alan Turing proves the correctness of a simple example program, again using assertions, but also considers how <i>loop variants</i> may be used to prove termination.
1961	John McCarthy proposes a <i>Mathematical Theory of Computation</i> which stimulates work on the formal semantics of programming languages (it is only possible to verify programs written in programming language L if L has a formal semantics).
1967	Robert Floyd's landmark paper <i>Assigning Meaning to Programs</i> introduces the <i>method of inductive assertions</i> and describes how a program annotated with first-order assertions may be shown to satisfy a pre- and post-condition specification - the paper also introduces the concepts of <i>loop invariant</i> and <i>verification condition</i> .
1968	A NATO conference is held in Germany to address the growing <i>software crisis</i> - the phenomenon that software tends to be delivered late and over-budget, and usually fails to meet the customer's requirements; this conference is regarded as the dawn of the <i>software engineering</i> approach to industrial software development and spawns the concept of <i>Computer Aided Software Engineering</i> (CASE) tools.
1968	Edsger Dijkstra's paper <i>Cooperating Sequential Processes</i> introduces the concept of <i>semaphore</i> to control the nondeterminacy inherent in concurrent programs.
1969	Tony Hoare's paper <i>An Axiomatic Basis for Computer Programming</i> describes a set of inference (i.e. formal proof) rules for fragments of an Algol-like programming language described in terms of (what are now called) <i>Hoare-triples</i> .

MMM and the SMVS Project

1969	Jim King's thesis <i>A Program Verifier</i> describes an early CASE tool constituting the earliest known implementation of Floyd's method of inductive assertions.
1972	Tony Hoare introduces the concept of <i>critical (program) section</i> , entry to which is typically controlled by a Dijkstra-style semaphore; critical sections are used by <i>concurrent</i> programs to control access to shared variables and other resources.
1975	Edsger Dijkstra's paper <i>Guarded Commands, Nondeterminacy and Formal Derivation of Programs</i> (expanded by his 1976 postgraduate-level textbook <i>A Discipline of Programming</i>) proposes that, instead of formally verifying a program <i>after</i> it has been written (i.e. <i>post facto</i>), programs and their formal proofs should be developed <i>hand-in-hand</i> (using <i>predicate transformers</i> to progressively refine <i>weakest pre-conditions</i>), a method known as program (or formal) refinement (or derivation), or sometimes "correctness-by-construction".
1977	Joe Stoy's <i>Denotational Semantics</i> is the first (postgraduate level) book-length exposition of the <i>mathematical</i> or <i>functional</i> approach to the formal semantics of programming languages (in contrast to the <i>operational</i> and <i>algebraic</i> approaches).
1978	Tony Hoare's (original) <i>Communicating Sequential Processes</i> (CSP) paper introduces the idea of concurrent processes (i.e. programs) that do not share variables but instead cooperate solely by exchanging synchronous messages.
1979	Robert Anderson's textbook <i>Proving Programs Correct</i> describes both Floyd's method of inductive assertions and Hoare's axiomatic (i.e. Hoare-triples) method.
1979	The <i>Stanford Pascal Verifier</i> project produces the earliest known formal program verification system to support a "real" programming language, i.e. Pascal.
1980	Robin Milner's <i>Calculus of Communicating Systems</i> (CCS) paper describes a <i>process algebra</i> permitting systems of concurrent processes to be reasoned about formally, something which for earlier models of concurrency (semaphores, critical sections, original CSP) has ranged from very difficult to basically impossible.
1980	Cliff Jones' textbook <i>Software Development: A Rigorous Approach</i> is the first full-length exposition of the <i>Vienna Development Method</i> (VDM), which has evolved (principally) at IBM's Vienna research lab over the previous decade and which combines the idea of <i>program</i> refinement as per Dijkstra with that of <i>data</i> refinement (or <i>reification</i>) whereby algebraically-defined <i>abstract data types</i> are formally transformed into progressively more "concrete" representations.
1981	David Gries' textbook <i>The Science of Programming</i> describes Dijkstra's weakest precondition method of formal program derivation, except in a very much more accessible manner than Dijkstra's earlier <i>A Discipline of Programming</i> .

MMM and the SMVS Project

1983	The 5-year <i>Alvey Programme</i> (with £200m of funding from the UK government and another £150m to be contributed by industry) is established (in response to the \$450m <i>Fifth Generation Computer Systems</i> project in Japan) to encourage joint industrial/academic research into <i>Intelligent Knowledge-Based Systems</i> a.k.a. AI ¹¹ . Software Engineering was one of the key focus areas of the Alvey Programme, with the particular goal of developing "Integrated Project Support Environments" (IPSEs), effectively integrated sets of CASE tools (performing various forms of program analysis, etc). One such IPSE to come out of the Alvey Programme was the <i>Mural</i> formal development support system for VDM. Overall, for all the money spent, the UK's Alvey Programme cannot really be considered to have produced any lasting AI-related results of genuine commercial significance.
1984	The Royal Signals and Radar Establishment (RSRE), Malvern, produces the MALPAS static analysis tool to assist the production of safety-critical software.
1985	Tony Hoare's <i>Communicating Sequential Processes</i> (CSP) textbook (currently the third most cited computer science reference of all time) presents an updated CSP model in which cooperating processes do not even have program variables and which, like CCS, permits systems of processes to be reasoned about formally.
1985	What will become MMM's SMVS project is born at SRL in Cambridge. The goal is to develop an <i>automated</i> program verifier V' (returning yes/no/maybe - and a definite yes/no for as high a proportion of real-world programs as possible), <i>not</i> a program verifier V (returning simply yes/no in all cases). The vast majority of Computer Scientists (at this time) do not fully appreciate the distinction, and therefore tend to brand the SMVS project as provably impossible (thus futile); it later becomes apparent that, until we have a finished product that people can play with, most people we might speak to about the SMVS (including potential customers) will either not understand what we are doing or not believe it to be possible, consequently MMM becomes the "dark matter" of the verification universe, developing the target SMVS product more-or-less in secret without publishing any results (in order to protect its competitive advantage) or any significant interaction with the rest of the computing world.
1986	RSRE's MALPAS tool evolves into SPADE and includes a verification condition generator; verification conditions (also known as "proof obligations") may be discharged using the SPADE Simplifier (an automated theorem-prover) and the SPADE Proof Checker (an interactive threorem-prover, used for those verification conditions that are too complex for the Simplifer to discharge automatically).
1988	Mani Chandy and Jayadev Misra's textbook <i>Program Design</i> introduces the <i>Unity</i> concurrent programming language along with methods for deriving Unity code from specifications - the first time this has been done for concurrent programs.

¹¹ the term *Artificial Intelligence* was distinctly out of favour in the UK following the 1973 Lighthill Report

MMM and the SMVS Project

1989	Mike Spivey's classic textbook <i>The Z Notation: A Reference Manual</i> summarises the formal specification language Z which, although originated by Jean-Raymond Abrial, has evolved (principally) at Oxford University over the previous decade (Z will ultimately become by far the most widely used language of its kind).
1989	Robin Milner's textbook <i>Communication and Concurrency</i> is a more accessible, although still technically advanced, exposition of his earlier CCS work.
1990	Anne Kaldewaij's textbook <i>Programming: The Derivation of Algorithms</i> and Carroll Morgan's <i>Programming from Specifications</i> describe mature methods for the formal derivation of algorithms (along with their proofs) from specifications.
1991	The UK Ministry of Defence (MoD) issues Interim Def Stan 00-55, mandating the use of formal verification when developing safety-related software for defence equipment. Given the prevailing state-of-the-art, virtually no-one can meet this requirement - it was in fact the MoD's intent to give the state-of-the-art a "push".
1992	After seven years of basic research, MMM's SMVS prototype is able to formally prove the correctness of "bubble sort" (as well as other algorithms of similar complexity) with all loop variants and invariants discovered automatically and all proof obligations (first-order theorems) discharged automatically. Few algorithms that industrial programmers actually write are as logically complex as "bubble sort" (which is why "bubble sort" is a classic computer science textbook example), consequently this milestone is judged as the "coming of age" of the underlying SMVS algorithms. Attention turns to evolving the <i>prototype</i> into a <i>product</i> .
1994	With the formal definition of the SPARK variant of the programming language Ada 83, SPADE evolves into the SPARK Examiner toolset (including the SPADE Simplifier and SPADE Checker), which is itself written in the SPARK language. The Spark Examiner is not particularly highly automated - end-users (who must necessarily be competent mathematicians) are required to provide all pre- and post-conditions and loop invariants, and must manually discharge many of the resulting proof obligations, limiting it to safety-critical applications. Nevertheless, the SPARK Examiner is the <i>only</i> mature tool allowing developers to meet 00-55
1995	Tony Hoare is MMM co-founder Aaron Turner's advisor at Oxford University.
1996	Jean-Raymond Abrial's <i>The B-Book: Assigning Programs to Meanings</i> (note the nod to Floyd's work!) is a long-awaited textbook description of his B method for the formal derivation of algorithms (along with their proofs) from specifications.
1998	Bill Roscoe's <i>The Theory and Practice of Concurrency</i> is a postgraduate-level textbook describing CSP (including mutually consistent operational, denotational and algebraic semantics) after many years of evolution at Oxford University.

MMM and the SMVS Project

1999	The previously sequential-imperative SPARK language is updated to incorporate those Ada tasking facilities defined by the <i>Ravenscar Profile</i> that are suitable for the construction of very high-integrity systems. Consequently, the SPARK Examiner may now be used to develop highly-reliable <i>concurrent</i> programs.
1999	Tony Hoare retires from Oxford University and joins Microsoft Research, who have been steadily recruiting world-class academics since 1991 in order to transform Microsoft into a world-class computer science research organisation.
2001	After a nine-year design phase, during which many competing ideas from mathematics and computer science were evaluated as potential components of the SMVS "jigsaw puzzle", and having also secured some Business Angel funding, source-level coding of the target SMVS product begins at MMM.
2002	The Z formal specification language achieves ISO standardisation.
2002	Microsoft Chairman Bill Gates remarks that "Software verification has been the Holy Grail of computer science for many decades" during a keynote address at the Windows Hardware Engineering Conference in Seattle; in the same year, Microsoft Chief Research and Strategy Officer, Craig Mundie, initiates the company's high-profile (and still ongoing) <i>Trustworthy Computing Initiative</i> .
2003	<p>Draft proposals for seven <i>Grand Challenges for Computing</i> are published, including <i>Dependable Systems Evolution</i> (ultimately known as the <i>Verification Challenge</i>) championed by Microsoft's Professor Sir Tony Hoare:</p> <ul style="list-style-type: none"> • the <i>Verification Challenge</i> is expected to take an international team of researchers around 1,000 man-years and cost ~\$1billion over 10-20 years • the (explicitly-stated) focus will be the pursuit of <i>scientific/technical ideals</i> • the <i>Challenge</i> aims to deliver a prototype (Alvey IPSE-like) toolset, the centrepiece of which will be an automated formal program verifier • the intention is to use the toolset to produce a million lines of verified code (at an effective rate of ~2,000 lines of provably-correct code per man-year) • it is not considered appropriate for an academic Grand Challenge to rely for its success on future theoretical advances that have not yet been made, therefore the <i>Challenge</i> will primarily implement and integrate various components of automated program verification theory that already exist • the <i>Verification Challenge</i> is <i>not</i> an imminent threat to the SMVS project's (estimated lower-bound) five-year technical lead, rather it completely legitimises the project and is a <i>total vindication</i> of everything it stands for • the Computer Science community is only now starting to realise what we knew 20 years ago - that an automated verification tool such as the SMVS is not only possible, it is the <i>only</i> possible industrially-practicable solution to the software industry's #1 problem - <i>getting the bugs out of programs!</i>