VISTA SOFTWARE LIFECYCLE EIGHT ESSENTIAL POINTS

FREDERICK D. S. MARSHALL

SATURDAY, 20 JUNE 2009

VISTA
EXPERTISE NETWORK

V I S T A

EXPERTISE NETWORK

VISTA SOFTWARE LIFECYCLE EIGHT ESSENTIAL POINTS

EXECUTIVE SUMMARY

1 · VISTA requires the VISTA software lifecycle.
2 · VISTA requires many code repositories, not one.
3 · Managing VISTA requires many authorities, not one.
4 · Users must directly control VISTA's software lifecycle.
5 · Users and programmers need a shared forum.
6 · VISTA's software stream requires many tributaries.
7 · We should restart the lifecycle with just File Manager and Forum.
8 · We need to declare interdependence and form a confederation.

I VISTA REQUIRES THE VISTA SOFTWARE LIFECYCLE

Over the thirty-two years of VISTA development, many, many different software lifecycles have been tried with it, and all but one of them have failed, sometimes subtly, sometimes spectacularly.

Interestingly, for the last fifteen years no one has consistently tried to follow that one proven model, and during that time every VISTA adopter has struggled with VISTA. Those who have deviated the least from the model, like Indian Health Service, have enjoyed the most success, and those who have deviated the most, like the Department of Defense (DOD), have suffered the most. Veterans Affairs (VA) makes the best test case to prove this point, since they have been at their most productive with VISTA when they followed the model, and at their least when they didn't.

From studying VISTA's rich and varied thirty-two-year history, I draw this radical proposition: we should take the plunge. We should end the fifteen-year drought by completely following VISTA's own software lifecycle model, the one that worked.

Getting to know the VISTA software-lifecycle model will take time, because it is sophisticated, complex, undocumented, and no example of it exists today, but I am confident that the more you get to know it, the more you will come to agree with me that the weird qualities of this model exactly support the weird qualities of VISTA in a way no borrowed or adapted model ever can.

Of the eight points I am going to make today, this is the simplest, the most important, and the least likely for anyone to believe. I wish that weren't true, because this is the crucial missed point at which most VISTA adopters went off the tracks since 1994. If you can resist the urge to "improve" a model you do not yet understand, if you can compel yourself to study it patiently as though it

were complex enough to deserve your attention, then you can buck the odds and reap the rewards that come with it.

II VISTA REQUIRES MANY CODE REPOSITORIES, NOT ONE

VISTA is bigger than you think it is, by more than one order of magnitude. None of the numbers you can put next to it (files, programs, lines of code, function points) comes remotely close to its actual complexity and sophistication. The interaction of its complicated integration with its unprecedented extensibility creates necessary and intricate effects we don't begin to know how to measure. Fortunately we don't have to.

We just have to understand that it is very, very big, too big to be effectively managed as a whole, unlike most software in the world. As a whole, VISTA is unmanageable, which is why in the VISTA software lifecycle, management of VISTA does not begin with or center around one, single software repository. That will not work. You will know you have begun to grasp the scale of VISTA when the idea of a single, central code repository for managing VISTA literally makes you laugh. Until then, you're not even close.

Fortunately, you do not need to begin with a single code repository to manage a shared VISTA code base. Really, you can't. VISTA will stagnate if you do. We never did, and the VISTA software lifecycle model doesn't begin there.

Instead, we follow the traditional divide-and-conquer strategy that makes computer science possible. We break up VISTA into packages and manage each package with its own independent code repository. In our experience, this reduces the scale of the problem enough that a highly expert, dedicated team can almost—almost—keep up with the problem of managing just that one package.

So, throw away the idea of beginning your VISTA lifecycle management

with a single, complete gold account, and replace it with the idea of many gold accounts, one per package.

Like all the points I'm making today, there's a lot more to this one than we have time to go into (for example, managing one VISTA package actually requires a group of VISTA environments be set up and maintained), but the basic idea should be enough to steer you generally in the right direction.

III VISTA REQUIRES MANY AUTHORITIES, NOT ONE

An unimaginable amount of expertise went into creating VISTA, and that same degree of expertise is required to manage it. Managing the VISTA software lifecycle requires more expertise than you have, more than I have, more than anyone has, but it also requires more speed and accuracy than any committee or bureaucracy is capable of. Any individual in charge of VISTA would be ignorant of at least 99% of the subjects needed to manage all of VISTA, and on any committee large enough to include all the experts needed, 99% of them would just be in the way for 99% of the decisions. There is no form of centralized authority capable of managing VISTA effectively, no matter how you divide up the code into repositories.

That is why VISTA's second name, before "VISTA," was the Decentralized Hospital Computer Program, because it cannot be managed well by any centralized authority, no matter how smart or powerful or well funded they are or how good their intentions. VISTA authority must be decentralized.

However, history has demonstrated that VISTA authority also has to be concentrated to ensure it is efficient and responsive enough to keep up with the changing needs of medicine, to avoid the chaos of a free-for-all, and to avoid the endless debates over trivia to which committees are all too prone. Each VISTA authority must have near-tyrannical powers over their chosen part of

VISTA to ensure maximum efficiency. So, VISTA authority must be concentrated.

VISTA also requires a third element in its governance. Each VISTA package is so complex it can only be effectively managed by dedicated expert programmers who can concentrate on mastering their chosen package over very, very long periods of time, like a decade or more. Most VISTA packages are too complex for any single expert programmer to manage, certainly too complex for dilettantes to manage effectively, so teams are required. And so, VISTA authority must be expert.

The resulting form of governance, to have authority both decentralized and concentrated into the hands of near-tyrannical teams of experts, is so weird there isn't a name for it—I call it the VISTA model of authority—but history has proven that whatever you call it it is capable of managing VISTA better than any other authority structure.

So if you decentralize authority, divide it up by subject and allocate it to teams of expert programmers, each team managing one VISTA package's code repository, with the senior experts in charge of each team, who is in charge of the senior experts? Who decides what the priorities are?

IV USERS MUST DIRECTLY CONTROL VISTA'S LIFECYCLE

VISTA is too important to healthcare to leave its fate in the hands of managers, programmers, politicians, or entrepreneurs. Really, only VISTA's actual users—nurses, doctors, pharmacists, lab techs, radiologists, and other medical and support personnel—know precisely how they use VISTA second-by-second every day, so only they can really know how it falls short in supporting their work. Of all the people involved with VISTA they are also the only ones directly contributing to patient care, which after all is the point of this entire enterprise.

Therefore, in the VISTA lifecycle model, although anyone can request VISTA changes, user requests trump all other priorities. Directly. Not as interpreted for them by managers, experts, committees, or other bureaucratic forms of user disempowerment. User requests are collected and used directly as the marching orders for the VISTA development teams.

Users are the fourth and most important element of the VISTA model of authority. It is the users who decide what each team works on, and not by sorting through or voting on priorities but simply by reporting the problems they are having and requesting bug fixes or enhancements. As the reports accumulate, the urgent problems identify themselves and do not require committees or expert panels to identify.

Many people find this element of the VISTA lifecycle idealistic, but since all of their proposed alternatives have failed over and over to the tune of billions of dollars wasted, and since this model not only worked for seventeen years but produced the greatest productivity and responsiveness the VISTA world has ever seen, I find the idea of doing anything other than user-driven VISTA development ridiculously idealistic. A properly organized VISTA lifecycle very nearly runs itself, using specialized tools and techniques of information management.

V USERS AND PROGRAMMERS NEED A SHARED FORUM

A user-driven lifecycle begins with teaching your users how badly we need their ongoing input about what works for them and what doesn't in VISTA. This is much harder for them today than it was in 1994 because Microsoft and other traditional software companies have taught them that no one gives a damn what works for them as long as they keep forking over the cash, that the programmers only work for them in some abstract, marketing sense. In our experience, it takes some work on your part to overcome this learned passivity, but

the users who break through tend to become excited chatterboxes, which is what we all need.

Next, we assume the majority of your users' VISTA requests are reported and resolved by the hospital's own Information Resources Management or Health Information Systems personnel, or by your immediate support organization's expert troubleshooters. The vast majority of problem reports are resolved with training, improved documentation, or through making minor local configuration changes to VISTA, and still others can and should be resolved with forward-compatible local extensions to VISTA. That is, a user-driven software lifecycle does not require you to air all or even most of your dirty laundry; each support organization handles the majority of its own problems, which after all is where it makes its support income. All of this can be handled through whatever local problem-reporting system you like.

But when it comes to problems that require shared changes to VISTA, problems requiring the attention of the VISTA package-development teams, VISTA problem-reporting cannot require the use of eighteen different systems for eighteen different organizations. We need to use a single, specialized problem-reporting system that has evolved over the decades to support our VISTA development teams, the one they are most comfortable and efficient with, the only one specifically designed to support the VISTA software lifecycle.

At the hub of the VISTA lifecycle is a single system called Forum. Forum is where users and VISTA teams carry on their continuous dialogue about how VISTA is working and not working for each user, where users formally make requests for changes to VISTA.

Forum is a VISTA system. It runs VISTA. Any VISTA system could be configured as a forum system, but we only need or want one to ensure we get all the user problem requests into one place where they can drive our development priorities. Configuring a VISTA system as the forum system mainly involves

ignoring all the medical, financial, and administrative packages in VISTA and properly setting up its communication packages. For example, one of the special VISTA packages Forum uses that most VISTA sites ignore is National Online Information Sharing (or NOIS), the main problem-tracking package; most VISTA sites do not use this package, but on Forum it is one of the main packages used.

We could spend all day just talking about how to configure Forum and how it works to organize user requests and encourage dialogue, but for today I just want to point out three observations.

First, someone needs to host this system and supply a VISTA-savvy system manager to run it. This is a responsibility, and it supplies a service that everyone needs, but there is no strategic advantage to being the one who runs it. You can't charge for it without screwing up the VISTA lifecycle, and you can't tamper with the flow of dialogue between users and programmers. It is a responsibility that carries with it shared benefit but no strategic advantage. At the moment, the network is testing out the Forum software on its Paideia educational server, and we could start out by using that until we're ready for a more robust system.

Second, because VISTA is medical software, sometimes problem reports must include patient information to properly diagnose. The only way this is legal is if all of Forum's users (programmers and users alike) have signed Health Insurance Portability and Accountability Act (HIPAA) agreements to respect the privacy of all patient information they see there. That is only possible if Forum is not open to the world, only to the finite community of actual VISTA programmers and users.

Third, Forum is not an optional or replaceable part of the VISTA lifecycle, though it is the first part that open-source enthusiasts get excited about replacing, usually the moment they hear the words "problem reporting." Everyone

wants to use either their home-brewed pet software or else the latest fad that is sweeping the open-source community. Such petty arguments over toolsets are at least half the reason why, eight years after I first proposed setting up a shared Forum system outside VA we still do not have one.

So this time, let's sidestep this problem by starting out with the only hub ever proven to work with the VISTA lifecycle. Later, after the lifecycle is underway and we are all productively developing VISTA we can tinker with our toolset, but for now, I refer you back to the first point I made. Let's try to suppress our urge to tamper with the VISTA lifecycle before we truly understand it.

The shared Forum accomplishes half of the VISTA lifecycle: it gets the development priorities properly set in response to user needs. After that, the second half of the lifecycle is easy.

VI VISTA'S SOFTWARE STREAM HAS MANY TRIBUTARIES

Each VISTA development team relies on user feedback in NOIS on Forum to figure out its top priorities, but the changes are not made on Forum nor on any central code repository, as discussed in point two. Instead, each package-development team develops its changes in its own package-specific code repository. Developers do not need permission to make changes to their software; they are the tyrannical owners of that software, and they continue to be just so long as they keep their users happy.

Once the developers have solved a problem, they bundle it up using the Kernel package's Kernel Installation and Distribution System (KIDS) and e-mail it to Forum. Using another special Forum package called the Patch Module, they convert the KIDS distribution into a patch and distribute it for testing, verification, and eventually release. The Patch Module maintains a list of subscribers for each VISTA package, so when a patch is released it is automatically

e-mailed to all of its subscribers. The power of this e-mail-based push mechanism is something we do not have time to get into today, except to say this is what makes the pending auto-patch system work.

Conceptually, this means the software stream does not flow to a single mouth, a single repository where people have to go to find upgrades. Rather, after Forum the software stream splits into a delta that delivers a stream of software to each subscriber.

This explanation should make clear that the patch stream, the nonstop sequence of incremental improvements to VISTA, also does not begin with a single source, but with many sources, maybe between fifty and a hundred. Forum's role on this outbound half of the lifecycle is strictly that of a traffic cop who sequences the traffic, not that of traditional software manager who controls what gets done and whether it can be released. Forum's role here is to act as the point where the tributaries come together to form a single stream.

Although from the VISTA adopters' perspective Forum produces the steady stream of small advances they can trust to continuously upgrade their systems, from the developers' perspective it is still very easy to separate out, think about, and manage just the thread of development that represents their chosen package. This is how the unmanageable scale of VISTA development is broken down into incremental steps organized into manageable packages and corresponding patch streams.

Although it hasn't happened much over the last fifteen years, a vital part of this software stream is that every year or two each package should release a fresh version of their entire package, an upgrade that goes through a more intense re-engineering, documentation, testing, and verification cycle than patches generally get. This helps keep each package's architecture fresh and provides a deep housecleaning period to flush out the subtler or more intractable bugs.

So, the "patch stream" is actually a software stream that includes both patches and new versions.

So what about complete snapshots of VISTA? Most software in the world is managed by releasing entire new snapshots. What about VISTA?

Well, this is part of where the VISTA lifecycle turns everyone's expectations on their head, part of why our software lifecycle does not begin with a single code repository. Nobody upgrades VISTA with a complete snapshot. VISTA's code and data are and must be far too intricately intertwined to simply swap out all the code like most software does. Instead, we upgrade with incremental changes, with patches and new versions of individual packages. At best, VISTA snapshots are useful for starting a brand-new VISTA system from scratch.

Even if VISTA were not so vast as to be unmanageable as a single codebase, this is the other reason we do not begin the software stream with a single code repository—because most VISTA adopters have little use for new snapshots. Instead of being the source of the stream, the VISTA platinum account (the clean and shiny VISTA codebase from which snapshots are created) is just another recipient of the software stream at the end of one of its deltas, like any other VISTA system. That is, the platinum code repository is a recipient, not the source, of the software stream. It has to be.

VISTA can only be managed piecemeal, with all its separate threads of user feedback and incremental changes being woven through Forum to create the fabric of VISTA.

VII RESTART THE LIFECYCLE WITH FILEMAN & FORUM

It has taken roughly 1,000 programmers driven by tens of thousands of users over thirty-two years to get VISTA to its current level of sophistication. To achieve a VISTA lifecycle renaissance will eventually require something similar.

But VISTA did not start on such a scale. Back when Ted O'Neill and Marty Johnson first launched what they called the MUMPS Systems (VISTA's first name), they only had about twenty-four programmers responding to a small population of users. That was plenty to create a system easily recognizable as the ancestor of today's VISTA, and it would be plenty to get things moving properly again.

To prime the pump, we should begin even smaller, with two small teams focused on two areas.

First, we should build on Medsphere's excellent File Manager work to create File Manager version 23. Fileman is the easy choice because it is the core architecture for all of VISTA and therefore the most important point at which to begin reinvigorating VISTA, since every VISTA package benefits from Fileman's improvements. That one package will be enough for us to restart, test, and refine the VISTA software lifecycle.

Second, in order to refine that lifecycle the associated software—KIDS, NOIS, the Patch Module, etc.—we will need to be able to fix or improve it too as we proceed with Fileman, so that software should be the other subject of our work, the focus of the second VISTA-development team.

This will require funding for the core developers for each of those packages plus the small constellation of students we need to surround each one with. It will also require funding for a Forum system manager and his students, a verifier and her students, and a database administrator and his students. The right fifteen to eighteen people could make this fly.

After the prototype lifecycle is up and running, we should plan to expand it to include the Kernel, Victory Programming Environment, and Laboratory packages, with a team of four to six people, mostly promising students working under the top gurus for each package.

Rebooting the VISTA lifecycle will also require the right license for the soft-

ware. We cannot use a license for the core infrastructure packages that discourages the widespread adoption of the common shared VISTA core, by adopters and developers alike. If that core is not kept in common, particularly the infrastructure packages like File Manager and Kernel, then VISTA will continue to balkanize into mutually unintelligible dialects, drifting apart as the VA and DOD dialects have done.

Although there may be VISTA packages for which the GNU Public License (GPL) is a suitable license, it is certainly not suitable for the core infrastructure packages because of the extreme degree of integration these packages have with all other VISTA packages, an integration so extreme it pushes past normal definitions of such terms as "derived works" and "interfaces." This unavoidable mismatch between the GPL's terminology and VISTA's unique internal structure makes eventual court battles over licensing violations highly likely, and offers no guarantee that such lawsuits would be decided on the basis of a reality most people do not comprehend and the rest cannot explain clearly. That is, although the GPL is not inherently incompatible with File Manager, for example, its terminology misrepresents Fileman's relationship to the rest of VISTA so badly as to make it an unreliable safeguard of our intentions.

Whether the license for the VISTA infrastructure packages needs to be the Lesser GPL, the Eclipse Public License, or some other open-source license is a question that needs to be sorted out in short order so we can get on with the work such licenses are intended to safeguard.

Most people should never ever work on the File Manager or Forum software because of how complex they are and because of the potential of problems in these areas to affect every other VISTA package, but they must be our top priorities. With the right experts leading these efforts, they will be the right projects to restart the lifecycle.

VIII DECLARE INTERDEPENDENCE & CONFEDERATE

Once upon a time, Ted O'Neill and Marty Johnson set into a motion the VISTA software lifecycle, which resulted in continuously improving medical software so popular with the users that the VA and congress reluctantly agreed to set aside their multi-million dollar failed boondoggles and do the right thing. For seventeen years the VISTA lifecycle made VA the main hub of VISTA development worldwide. With VA providing such a reliable heartbeat for that lifecycle, everyone else could afford to remain comparatively disorganized so long as they stayed in a symbiotic relationship with VA and its lifecycle.

The future before us is very different. With VISTA spreading all over the world at an increasing pace and VA's mandate restricted to healthcare for veterans, VA will shift from the center of the worldwide VISTA lifecycle to become a peer. That will require a radically different form of relationship among the various VISTA-interested organizations, one in which the heart of the lifecycle does not exist within a single organization but across multiple, sometimes competing organizations. This form of relationship has a formal name. It is called a confederation. The future of the VISTA community is a worldwide confederation.

The present, caught between that past and this future, demands a time of radical but wise change. Now is the time for the transformation of our community. Now is when we must breach the barriers that separated us into autonomous feudal kingdoms, each with its own dialect of VISTA. Now we must declare our interdependence and forge the formal alliances that will pave the way for confederation, supporting our need to compete and innovate while still collaborating on the shared lifecycle and software upon which the health of our patients depends.

The world has begun to marvel at VISTA, to realize the amazing power it has

to help doctors and nurses save lives. But we know they haven't seen anything yet. We know the real power is in the software lifecycle that created VISTA, a lifecycle we are about to unleash again to work its miracles for us. We know the world is in for a wonderful surprise.

HISTORY

VISTA Software Lifecycle: Eight Essential Points
Saturday, 20 June 2009 · VISTA Expertise Network
Frederick D. S. Marshall
First Version
http://www.vistaexpertise.net/docs/VISTA-Software-Lifecycle-speech-vi.pdf

COPYRIGHT