Under the Hood:
Open Source Business Models in Context

Stephen R. Walli

Vice-president, Open Source Development Strategy

optaros

Notes: Introduce my background. When I say open source I mean free and open source software, I understand the difference, and assume you do too. We will talk about patents at several points, and to be clear, I don't believe software patents are particularly useful.

There is [still] huge confusion over the OSS business model today

Notes: Everyone wants to categorize things. Is it an OSS company? Closed source? Really a services play or a hardware company? Maybe it's a hybrid-OSS company. These are the wrong questions. It ignores the fundamentals of a business with a value proposition to a customer.

We continue to think OSS is “New” and “Special”

Notes: The discussion degenerates into political rhetoric and zealotry.
We can’t apply “normal” rules around a business for success because we think we need new rules. Not unlike the Internet boom-and-bust.
I am talking about business here – not communities, or the value of communities – but businesses that grow up around the community or are associated with the community. .

This makes our businesses susceptible to the FUD of larger organizations and holds back larger enterprise (customers and vendors alike) from participating

Economics Works:
OSS communities and businesses demonstrate the same economic behaviours as any other human social endeavour in a marketplace

Notes: We have known how communities and economics worked since You had a campfire and I wanted to sit beside it. Before the bazaar and the marketplace, I would argue that that campfire was the first economic transaction. As Bob Young pointed out, “A community is just a collection of people with something in common – they don’t have to like each other.”

Understanding the context of OSS in business will allow us to grow businesses faster using well understood business tools practices

Notes: Again: We are not talking about turning our communities into businesses … but rather when there are businesses around code communities we understand how that should work.

First let’s talk about standards …

No one comments on standards and commoditization

Notes: This one fascinates me. We've had standards in our industry forever and no one comments on the commoditization effects of standards.
We hear, "OSS is going to destroy the value of software", but standards appear to be "business as usual".

Standards exist to encourage multiple implementations

Notes: Standards are about encouraging multiple implementations. As r0ml recently pointed out, you are giving up features to gain choice of vendor. A proprietary spec is by definition the opposite. It's about encouraging things to attach to the other side of the interface to develop a rich ecosystem of add-on products.

Don’t confuse de facto technology with de jure standards

Notes: De jure standards – there’s a process to create, participate, ensure fair discourse, prevent anti-competitive practices, and reap old work.
It could be a government based organization, an industry or trade organization, or a consortium of companies. There is a narrowing of focus/expertise as you travel that spectrum. Standards are a tool to commoditize a segment of a market to enable the growth of the market overall. I believe standards should be based on existing practice and experience … and we’ll see why.
De facto technologies are market dominant products. They reduce choice.

Patents exist to protect a single implementation – by definition they serve a different need on the economic spectrum from standards

Notes: It is the same problem as OSS and patents … one is encouraging multiple products or uses while the other protects a single use.
There are rules in every standards org on how to deal with patents when you find one in your midst. The new OASIS rules are actually exceptionally good – talked about why.

Open Source Software is “just software”

Notes: When r0ml and I have discussions around “open source” lately, we often ask the simplifying question up front. “This would be different from commercial software how?” It’s not that there aren’t differences, but it gets one away from thinking everything is different.

Good software is developed by good software developers – Quality is a function of community and leadership – distributed community forces discipline

Notes: Since 1980 I have had the privilege of working with some brilliant software teams. The best teams all had at least one and generally two completely anal software disciplinarians that don't know how not to build software without reading every line of every checkin, automating the build, doing nightly builds, automating the test environment, capturing the decisions in some form (email, web pages, and wikis), doing full config mgmt over version control. etc.

Why are there 90K projects on sourceforge but only a handful of truly successful ones? I believe the distributed community forces a hard line on the project – be disciplined or die.

There are a number of papers done over the last few years that have demonstrated that code inspection of ANY kind (formal or informal) finds more bugs than test. .

The software architecture reflects the componentized, composible, well defined interfaces of its UNIX heritage

Notes:When we get to Christensen we'll talk about this some more, but the idea here is that there is a place for monolithic stacks, but once a marketplace matures and the incumbent over delivers then componentization takes over. Also: disruptive business models start when someone takes inexpensive off-the-shelf parts and assembles them into underperforming solutions compared to the "norm" but these items find a new niche and begin their own sustained innovation growth as a technology.

UNIX has always been a modular architecture, there are standards around it, and this is reflected in the software communities that started there.

The ability to build LAMP stacks is very powerful when you're a startup or when you need a new product quickly to complement other offerings.

Participation reflects a normal asymmetric value proposition – you get more than you give

Notes: It's not about money. It is about economics. I get more than I give. When I work on Apache and give back bug reports or bug fixes or new functionality, and I download the next rev, I get a 100 times more back than I gave.

Marten Mickos recently said: "The early community is willing to trade time to save money. The late community is willing to trade money to save time." (LinuxWorld, Feb 2005)

It's not about altruism. People value their skill sets differently in different contexts: A writer may be a marcomm/tech writer during the day, helping a child with a school writing project, teaching an ESL class in the evenings, and writing a sonnet to a loved one. In every case they're a writer. In each case they're valuing their skill set differently. It's about the economics at that first campfire level.

The great thing we'll see (hopefully) is that it also works for company participation.

Open Source Licensing is ultimately what makes it “Open Source Software”

Notes: I DON’T go into a long discussion about licensing in this talk. The audience already knows more than is necessary.

It’s not a stack. It’s a network.

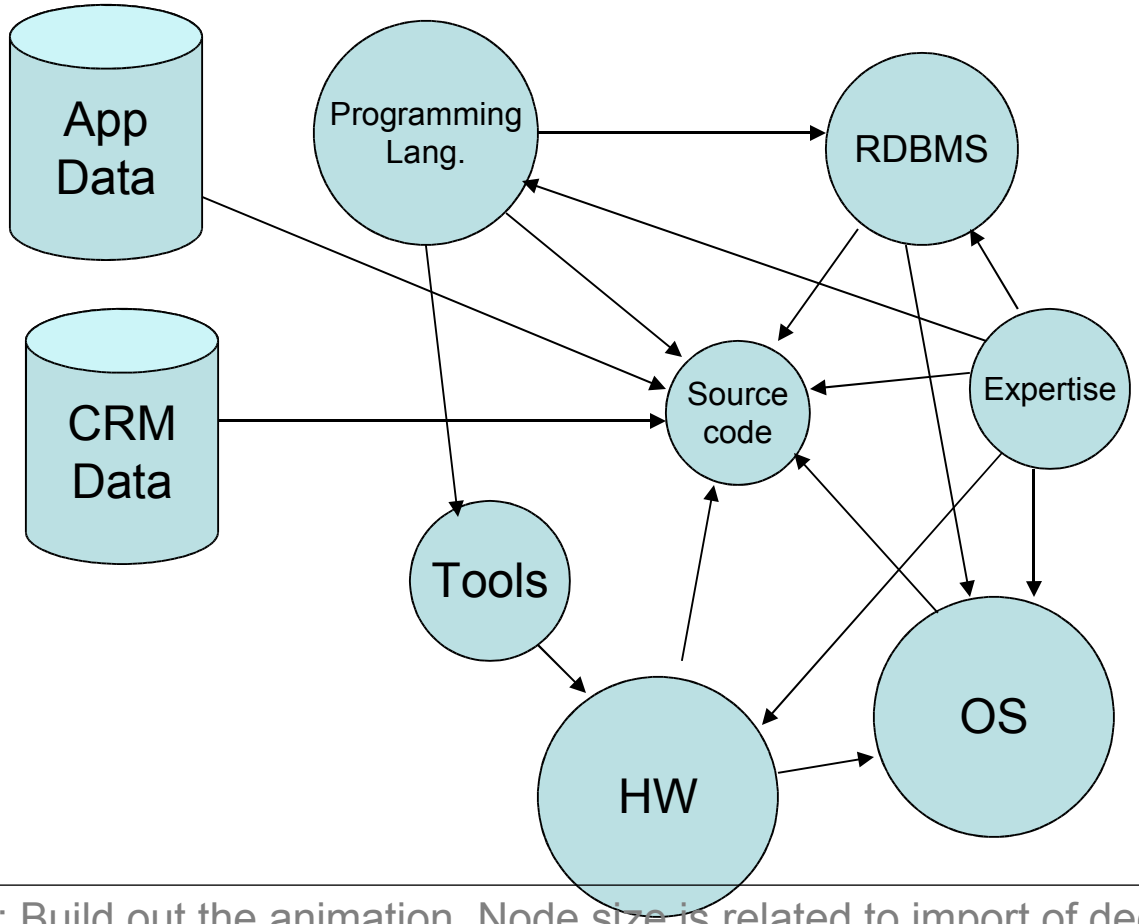
Notes: Everyone wants to have the discussion around how “Open Source will eat it’s way up the stack until software becomes valueless.” Getting away from the stack metaphor actually allows us to see the opportunities and challenges better. It gives us flexibility in our thinking.

Customers have a network for their solutions
Vendors have a network for their products

Business is about matching networks

Notes: Computer people understand networks as a mathematical concept and immediately seem to grok this metaphor. This discussion worked well at USENIX. MBAs and lawyers don't have the same understanding of networks and don't necessarily "get it", based on old presentations I've given to them. (Oddly enough Christensen does use "networks" of products in the first book to advantage.)

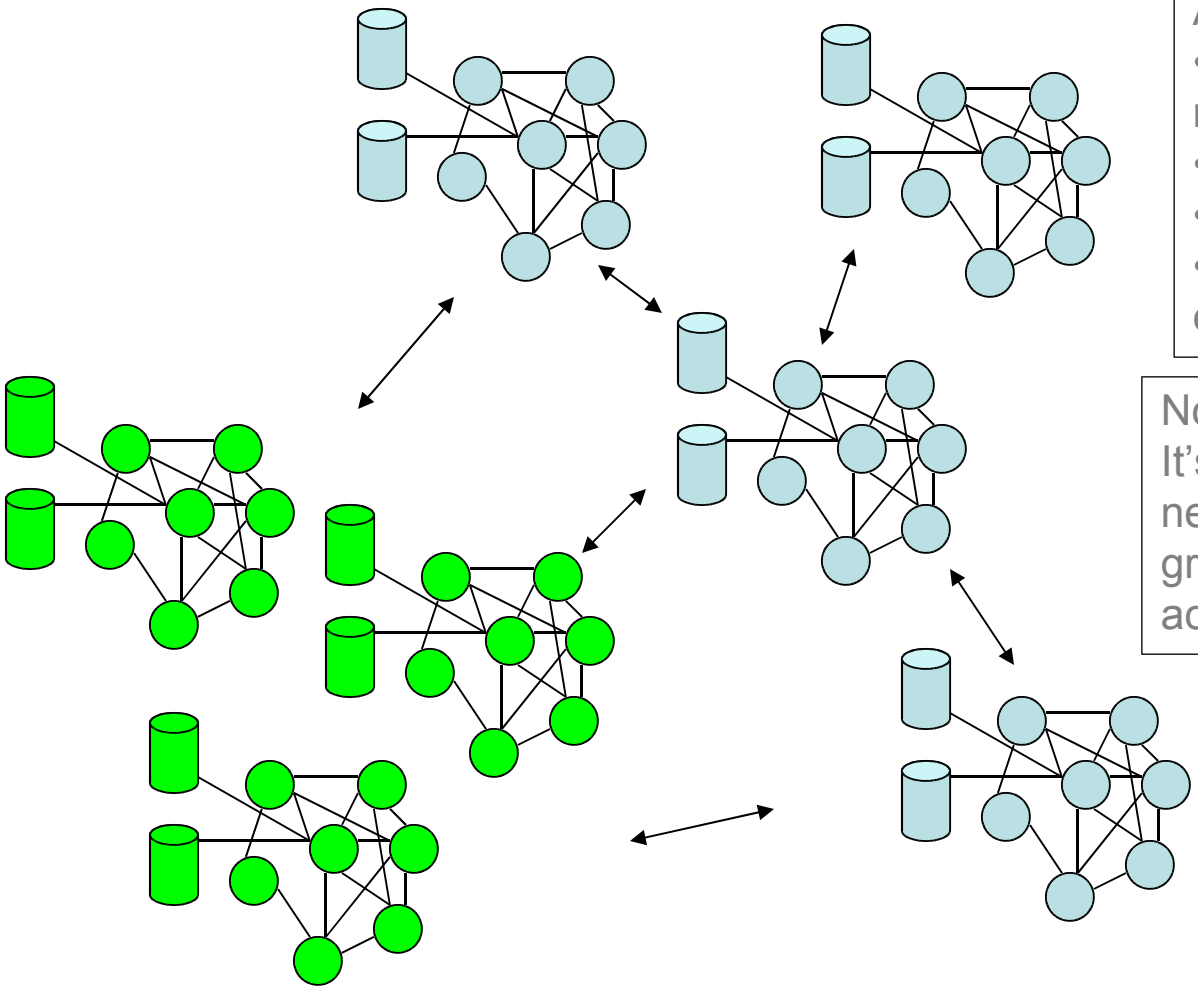
The enterprise application is actually a network …



Animation:
- start with source code
- add the other bubbles
- add the data stores
- add all the edges

Notes: Build out the animation. Node size is related to import of decision to enterprise group. Node size is “arbitrary” – every enterprise weights these decisions differently. Use the PL/I examples: forced to PL/I twice because that was the value placed by the IT dept. on IBM history/expertise over the DEC projects.

… and the enterprise application network isn't "simple"

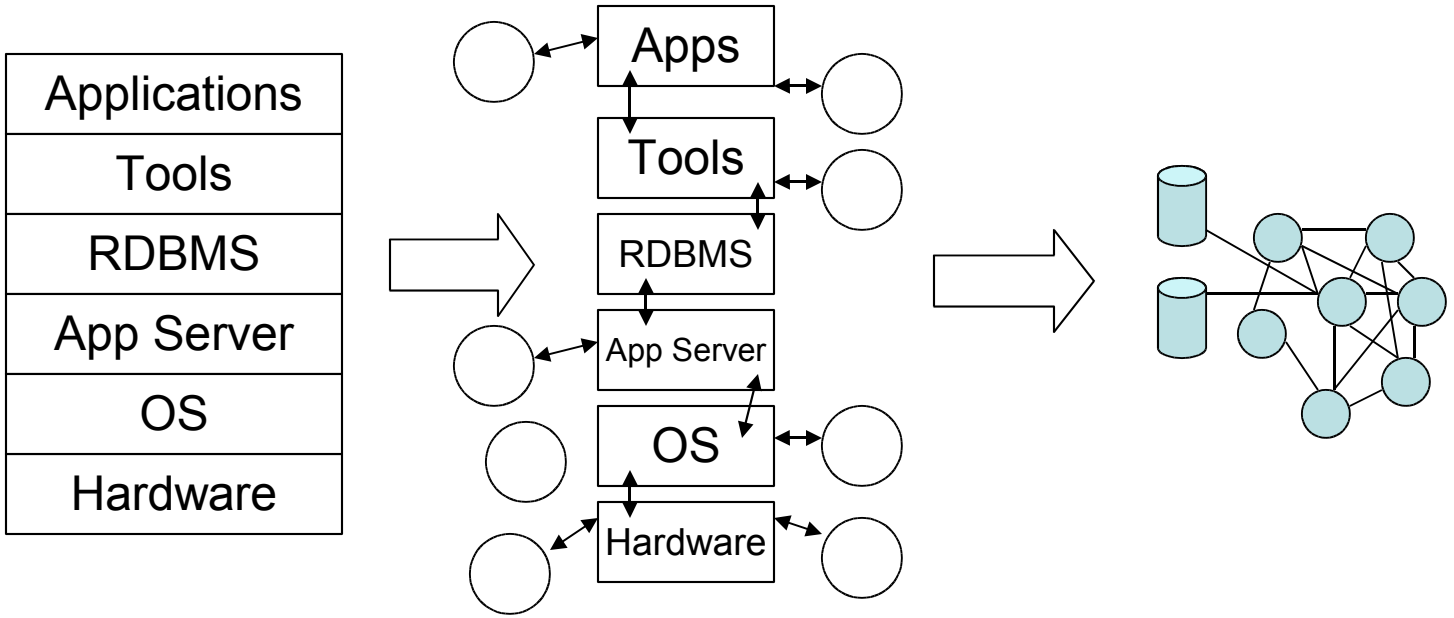


Animation:
- start with one network
- add other blue nets.
- add green nets.
- add all the internetw edges

Notes:
It's a network of related networks. I use the green to represent an acquisition.

Animation: 1. Start with left side “stack”. 2. Add the middle split stack. 3. Add the bubbles. 4. Add the right side network.

It’s not a stack – it’s a network.



i.e. the “stack” is a view through the network.

Notes: When vendors talk about OSS eating it’s way up the stack, they might better be relating where they see themselves in the food chain. A “stack” is merely an ordered viewpoint through the network. For example, a chip manufacturer has a very different view of this “stack” -- invert it! The value is in the hardware! This loosens up the thinking … fewer restrictions on the way you view the challenges and opportunities.

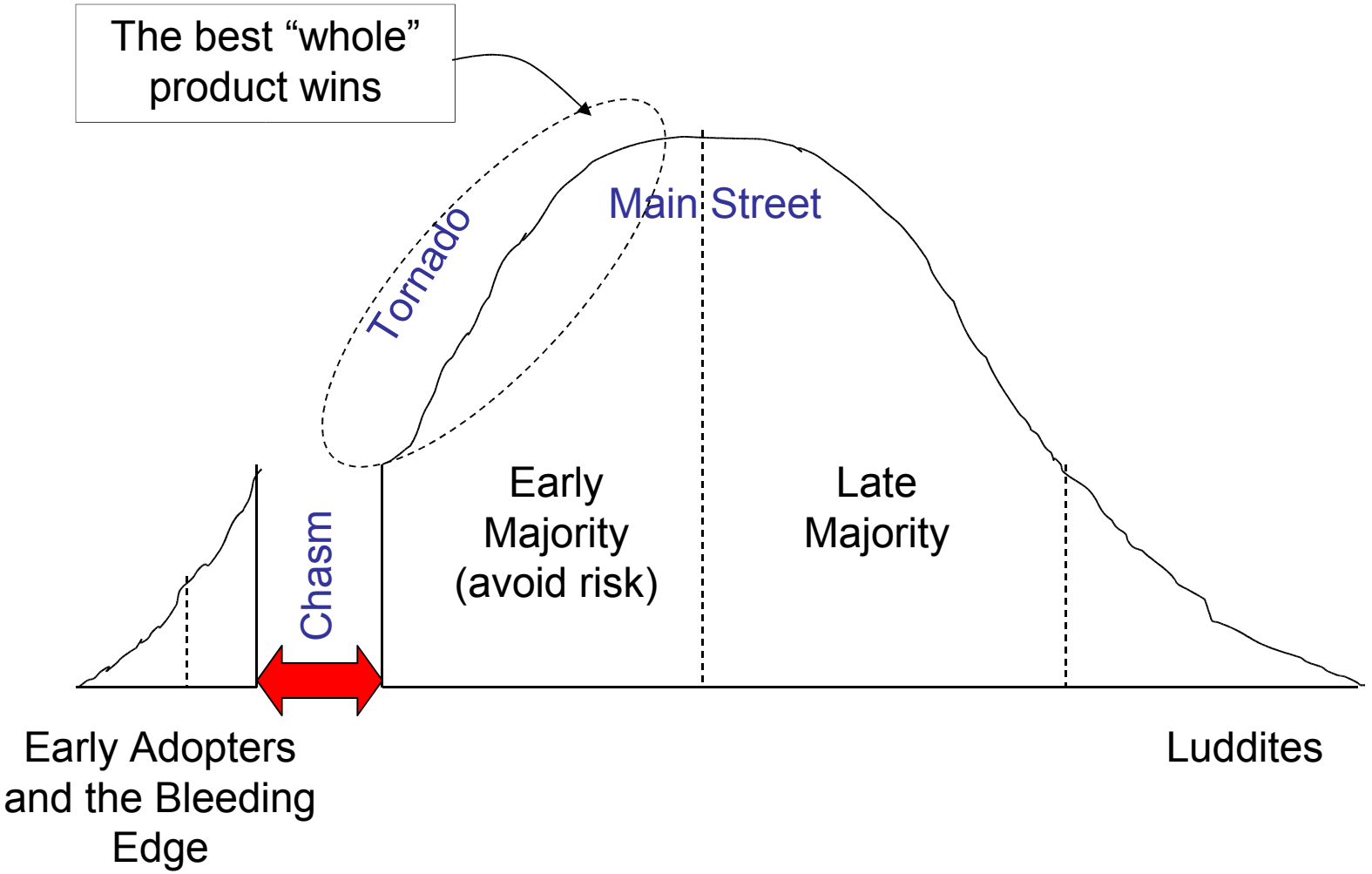
Customers develop procurement “standards” based on the architecture of their “networks” to simplify the complexity (i.e. to reduce choice)

Let’s shift gears for a moment to the vendor network …

Geoff Moore's Technology Adoption Life Cycle

Geoffrey Moore, Crossing the Chasm, HarperBusiness, (Revised) 2002

Geoff Moore's Technology Adoption Life Cycle (circa 1991)



Notes: Walk people through the simple version of the Technology Adoption Life Cycle. Talk about the risk profile in the early majority. Use the B&N and Win2K beta example around CIOs. The company that wins in the tornado market provides the best "whole" product.

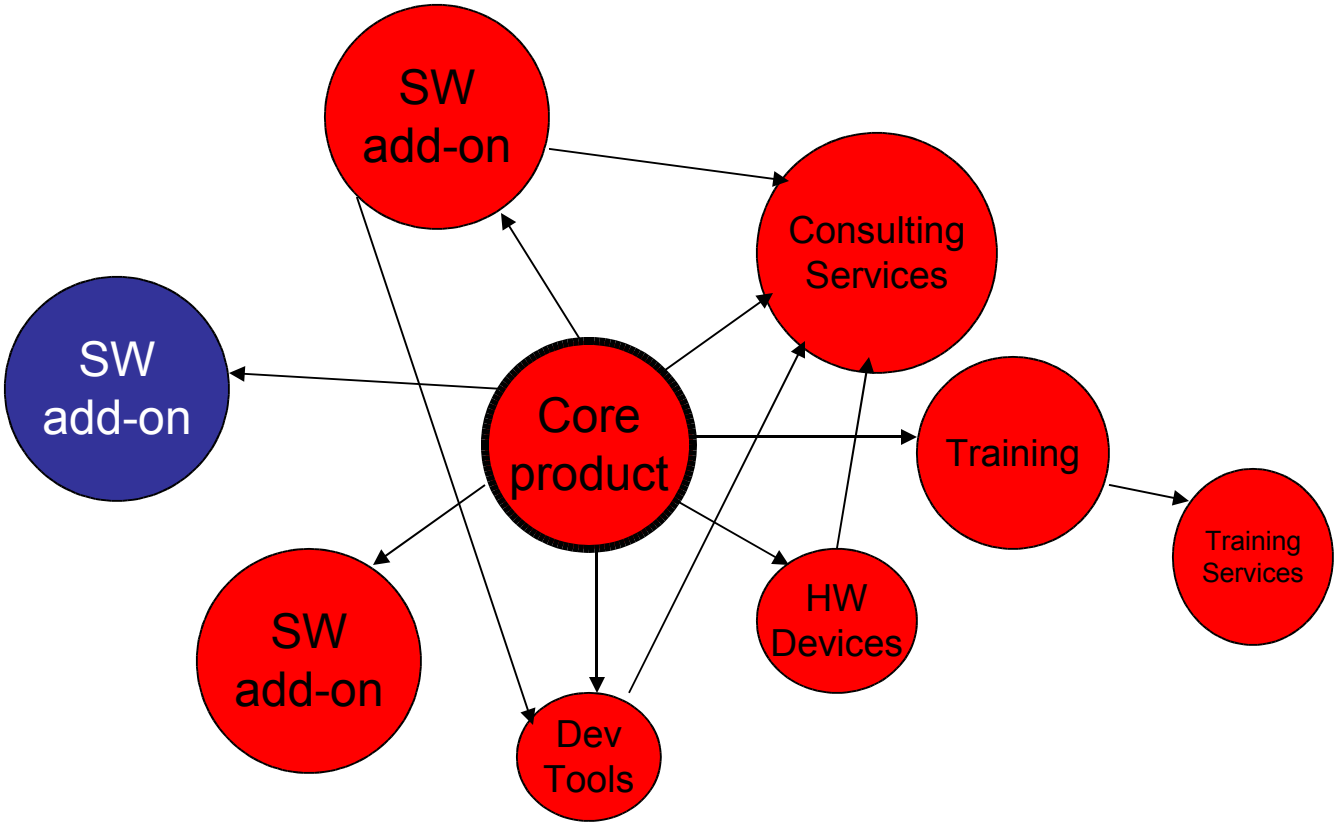
It’s about providing the core value proposition plus all the complements you can reasonably provide

whole product = core + complements

The different ways to develop and drive complements

- “Buy” or “build” and bundle
- Training programs (and train-the-trainer)
- Certifications
- Consulting services
- Publish interfaces to add value to the ecosystem
- Development tools
- Partnerships with other companies
- Hardware appliances

The whole product reflected as core and complements i.e. the network again from a vendor's perspective

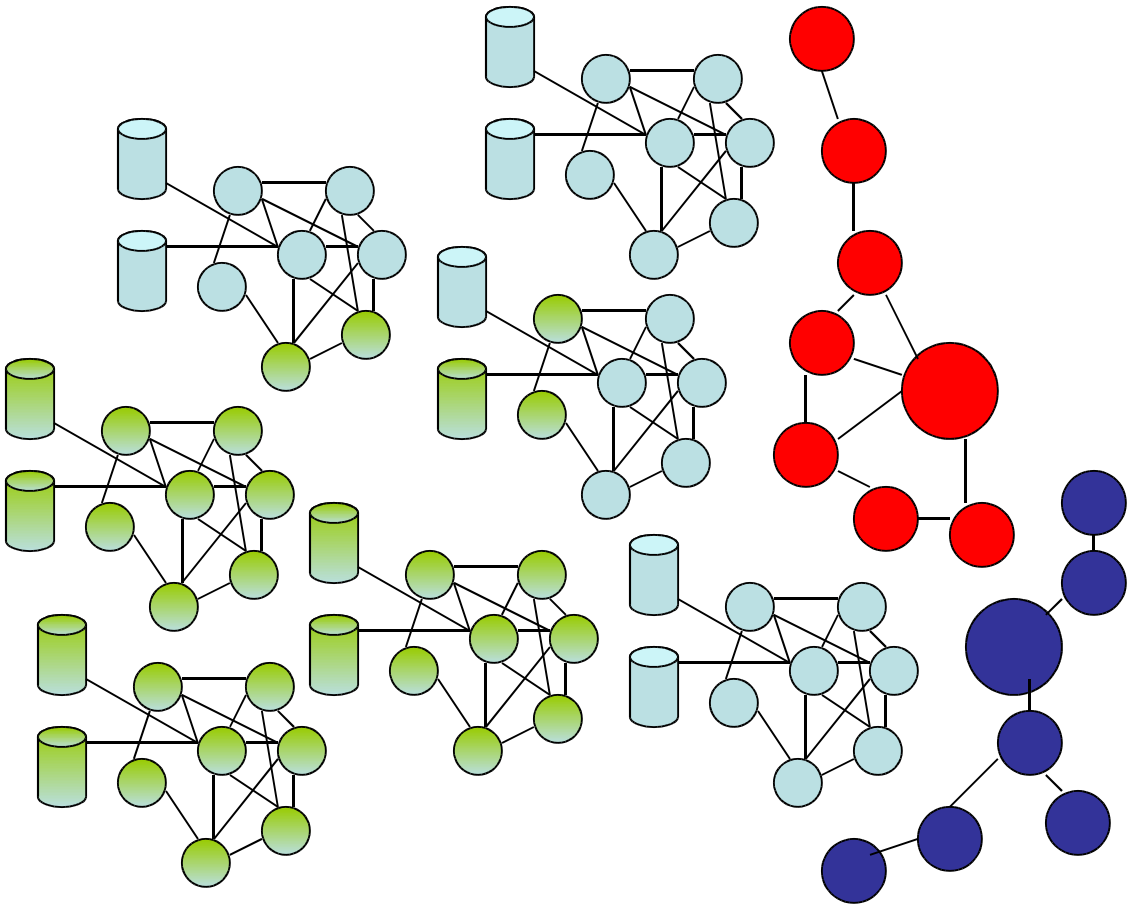


Notes: Blue bubble represents a partnership with another company.

Another way to think about this is the vendor wants to provide as many complements as they can to their core product offering, covering as much of the customer's solution network as is feasible to present the best (most valuable) solution in the customer's eyes

Notes: As long as the sum of the revenues across the network is greater than the sum of the costs/investments, the company remains profitable.

Business is all about matching networks …

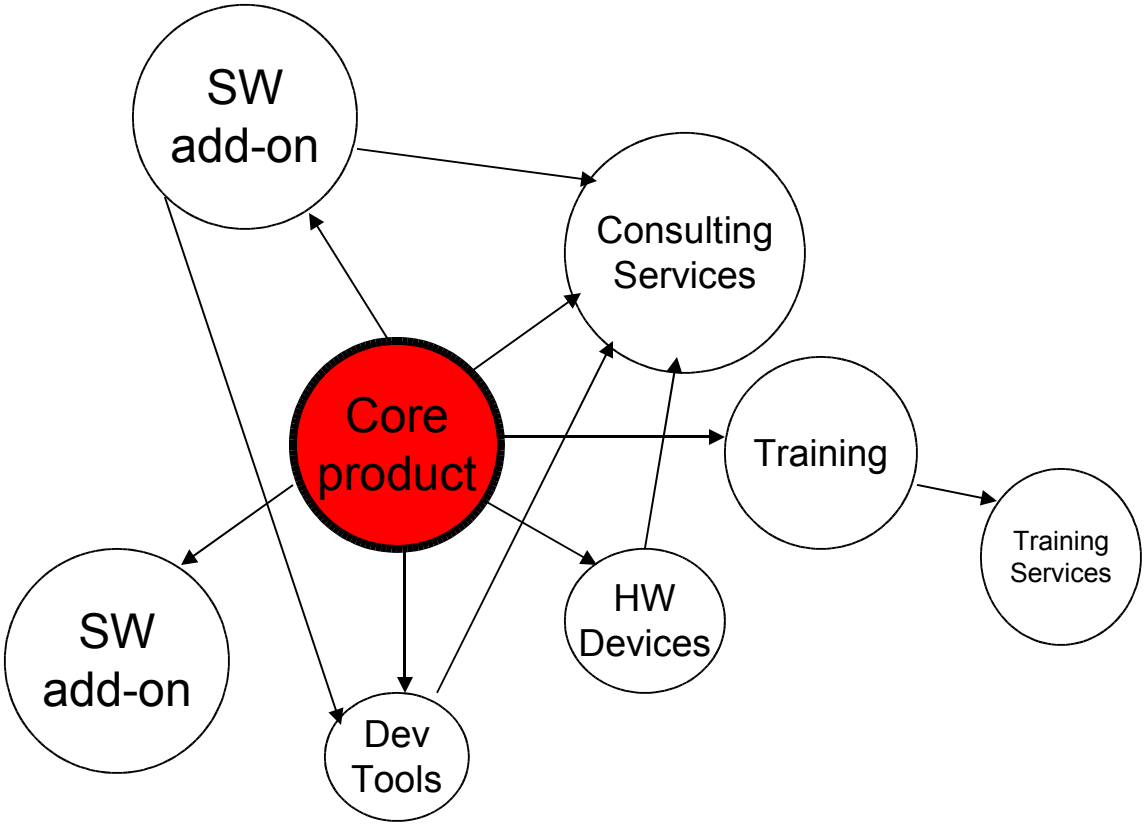


Animate: 1. Start with the enterprise networks. 2. Add red vendor network. 3. Add blue vendor network.

Notes: Different vendors have different product network pitches, emphasizing different aspects of the product offering – indeed they must differentiate or look just like the competition (e.g Redhat and security, vs. SuSE and management.)

This is why different vendors create standards around nodes they each might share in their network while competing on the rest of the offering network. This is why a vendor might get testy when a standard appears on their core revenue stream node but other vendors are happy to participate in standardizing a shared complement.

From this network view we can apply a more sophisticated IP strategy



Intellectual Assets:
Ideas, Designs, Implementations, Services, etc.

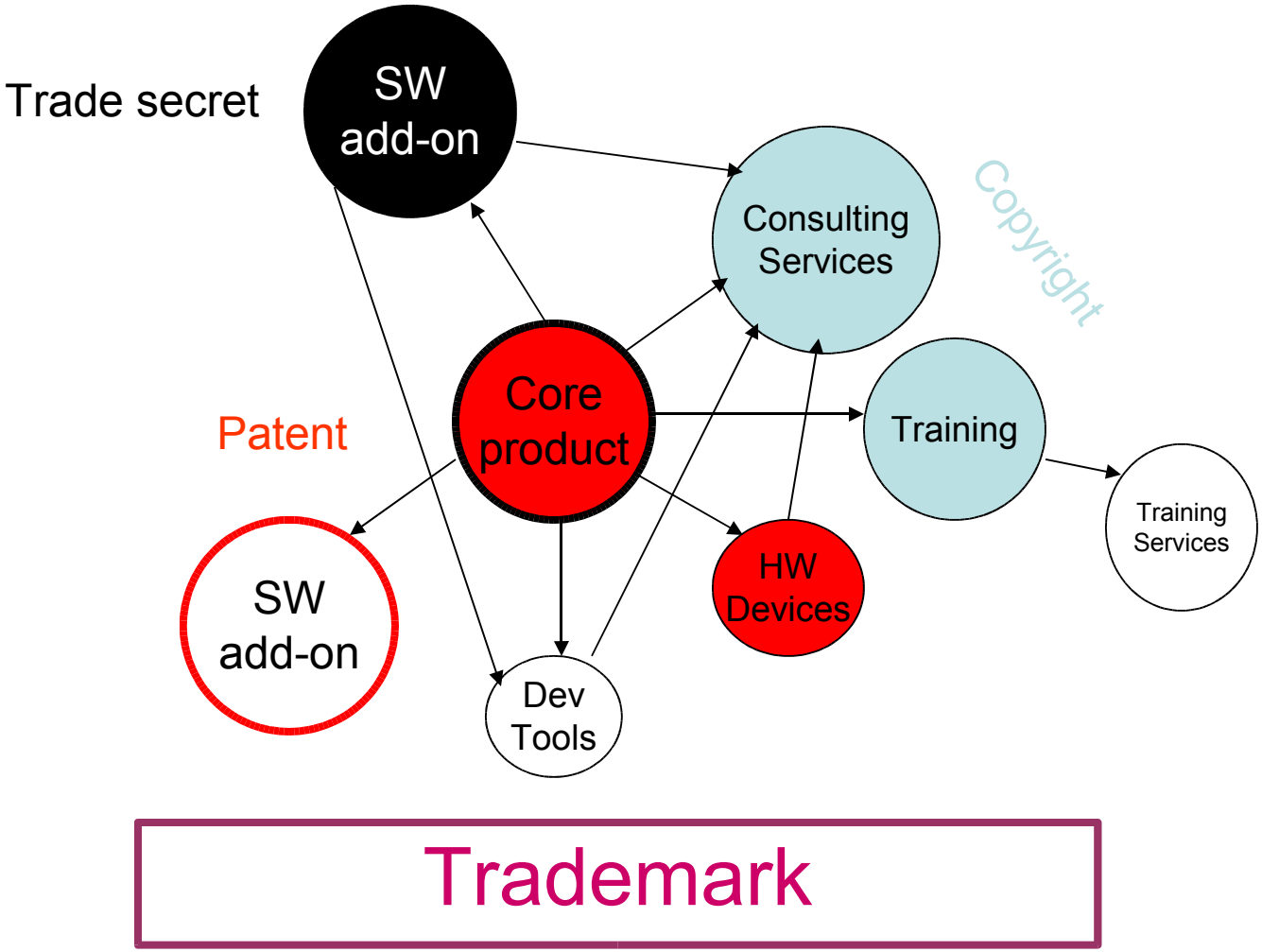
i.e. What your customer cares about.

Intellectual Property:
Patents, Copyright, Trade Secrets, and Trademark

i.e. tools that turn assets into legal property.

Notes: Lawyers are also sticklers for language; they just happen to program in English. “IP” has a very particular meaning to them. Distinguish between assets and property. Think about the entire product offering network and what’s core vs. complement – then think about an asset strategy and how best to use property.

Thinking about the IP tool investment



Notes: Discuss lots of ways to “colour” the nodes of the “whole” product. Consider all of them.

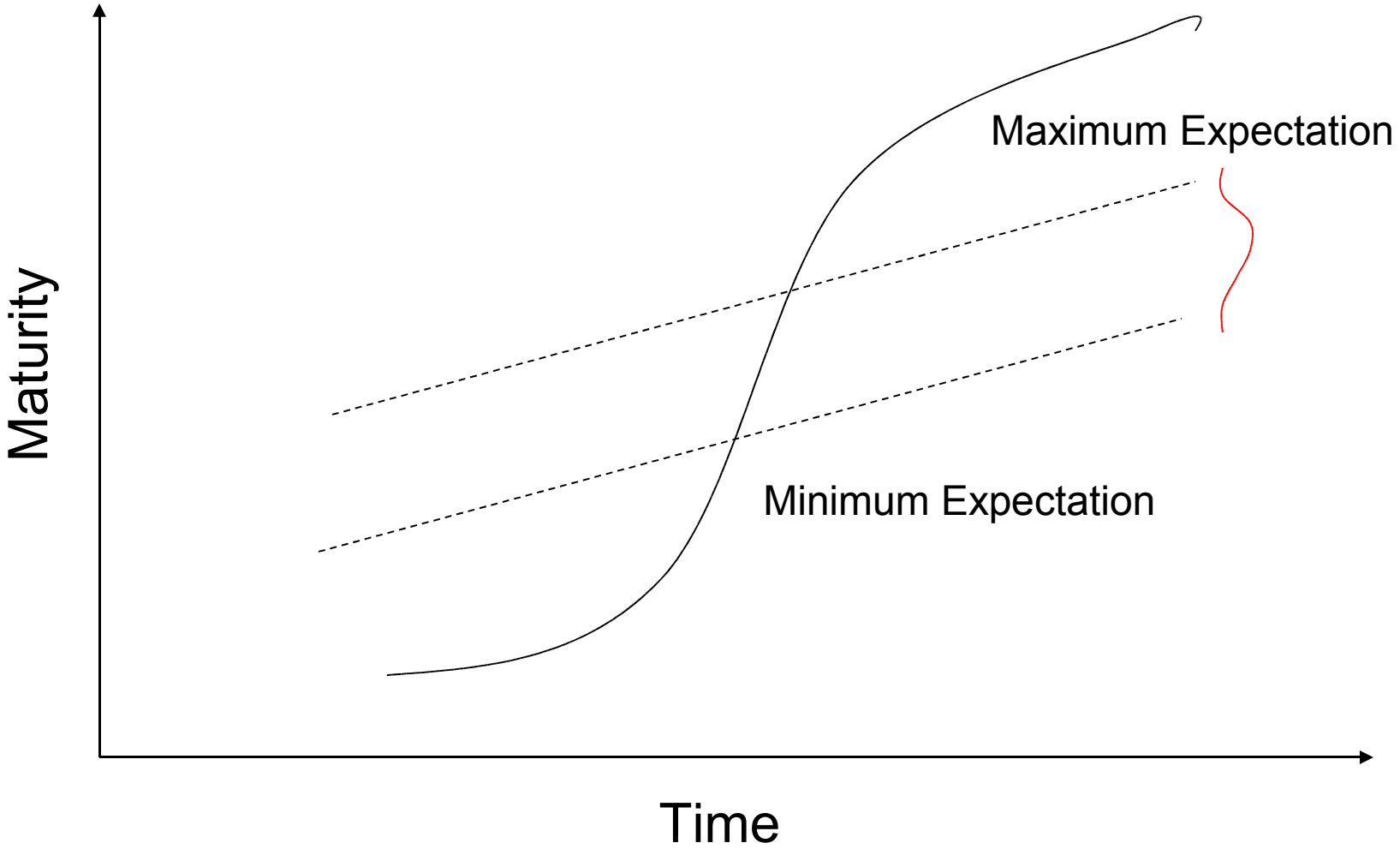
Publishing and other field salting techniques

Notes: Everyone wants to get patents. VCs push and valuate. Lawyers make money. But a small company is still lost against the big company portfolios. A patent is merely a ticket to the negotiation. However, laying down a pattern of “prior art” may help prevent the big company problem. Salt the fields around certain components in the network. Source code is just another publication medium. Discussed the (likely apocryphal) example of the IBM Systems Journal as a publication tactic.

Christensen and the Innovator's Dilemma

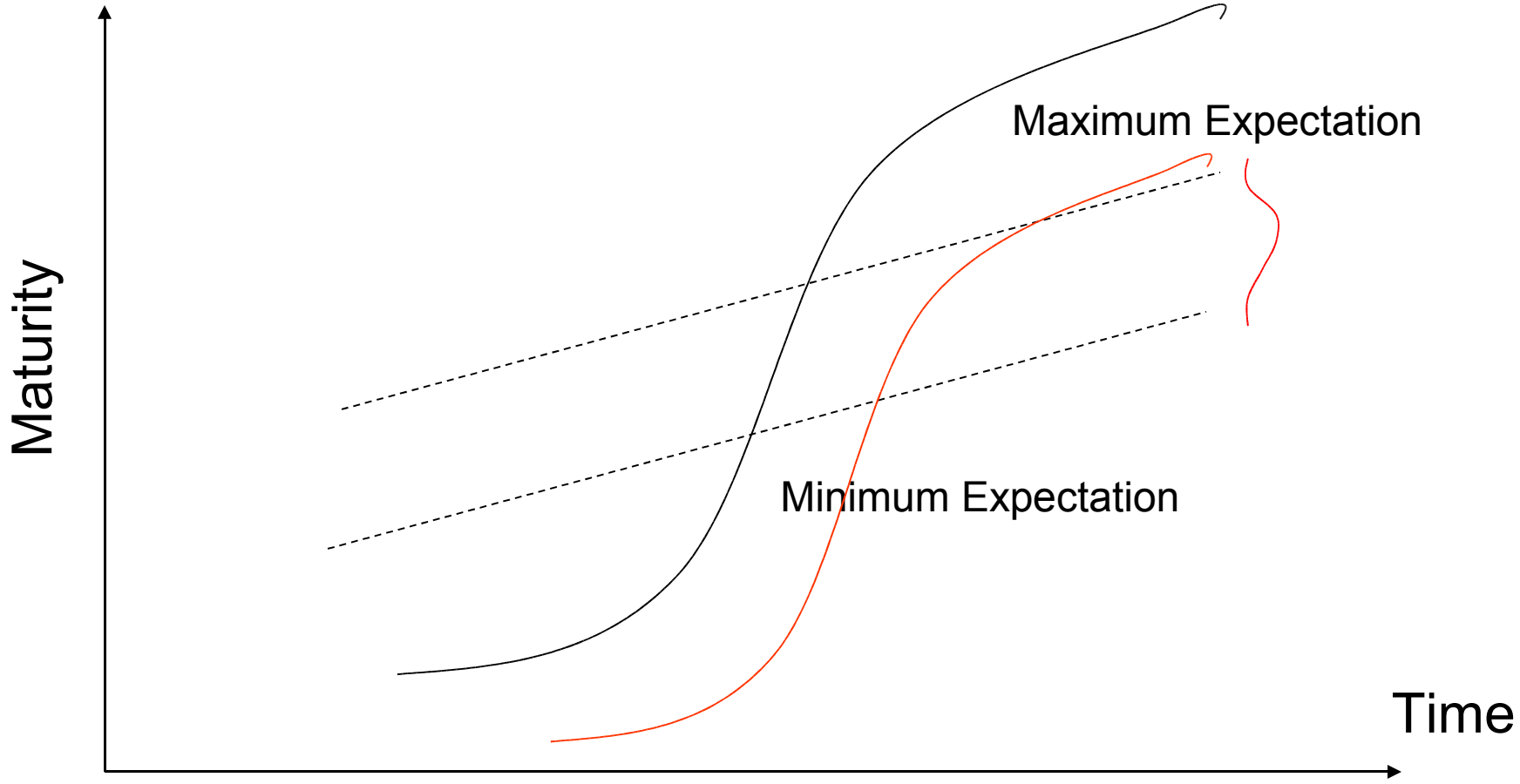
Clayton Christensen, The Innovator's Dilemma, HarperBusiness Essentials, 2003
Clayton Christensen, The Innovator's Solution, Harvard Business School Press, 2003

Innovator's Dilemma



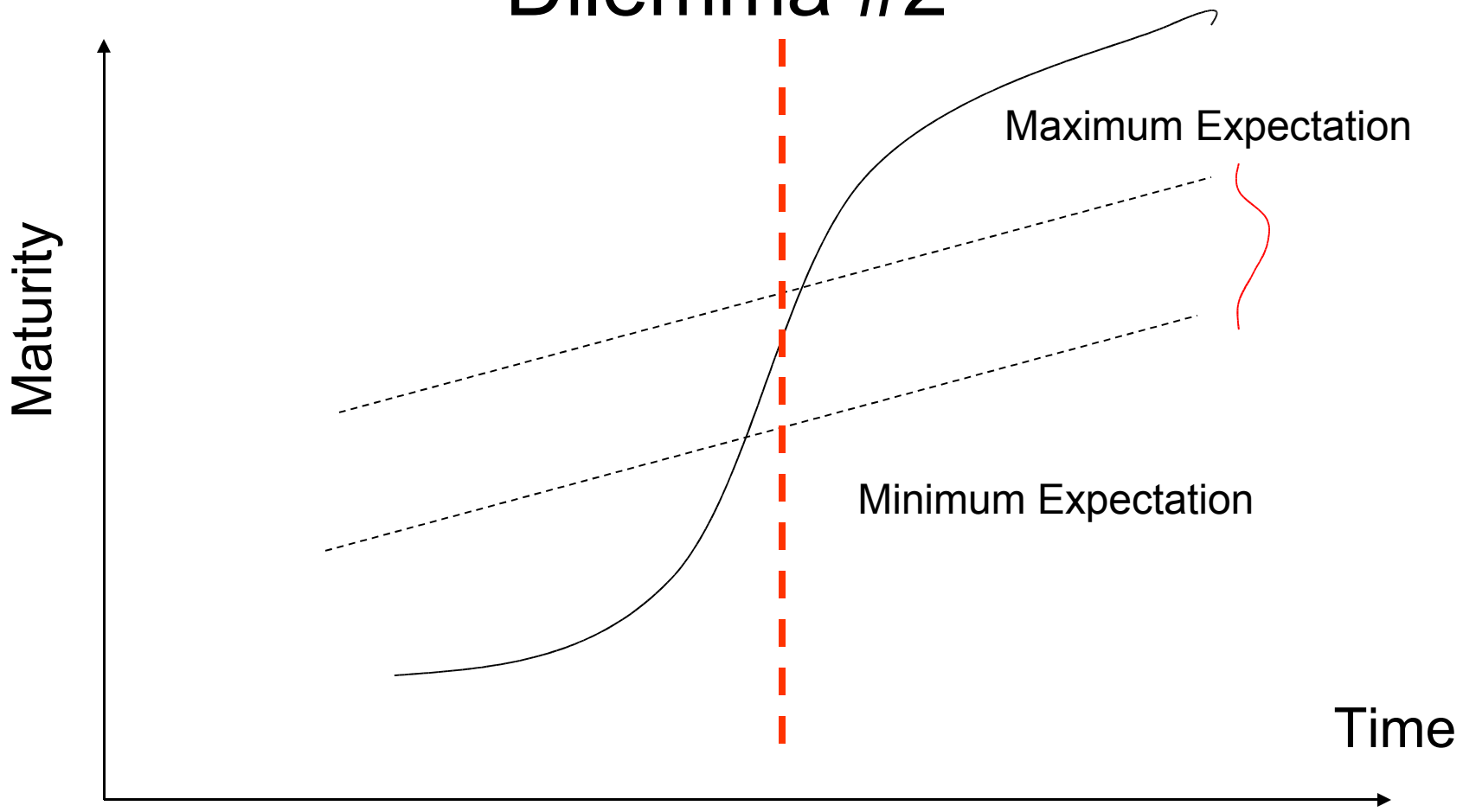
Notes: A basic walk through the primary thesis of Innovator's Dilemma.

Dilemma #1



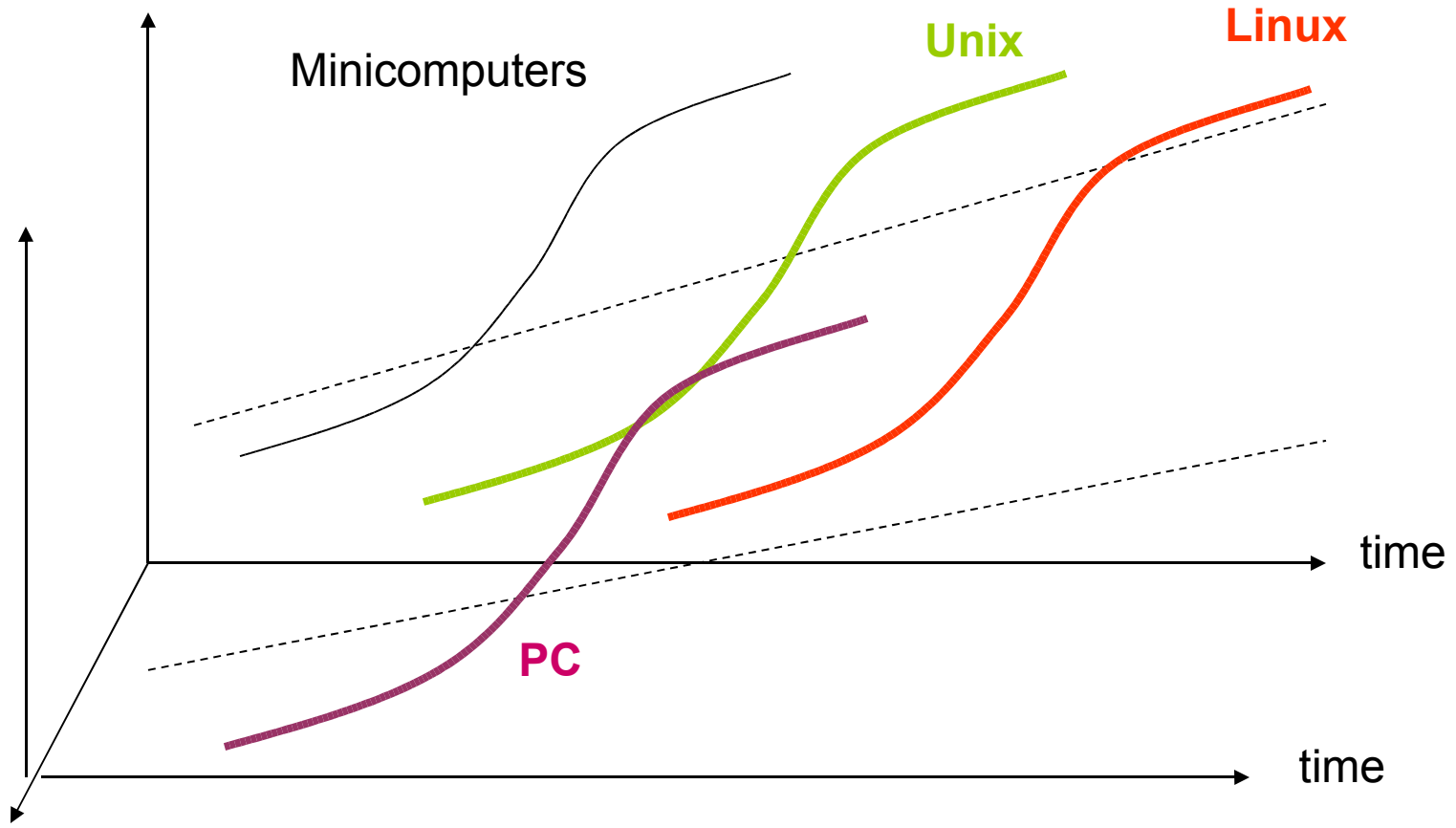
Notes: It's not a disruptive innovation; it's a disruptive business model (per Christensen's story about Andy Grove's observation). Walk through the basic premise using disk drives, and steel mini-mills, and the retreat up-market. These things are assembled from off-the-shelf parts and under perform BUT they find their niche then mature into the other product's space. Use Linux example as a teaching OS becoming something too complex to teach BUT it now makes a great business server in scaled out farms.

Dilemma #2



Notes: When you start delivering faster than your customer can absorb technological innovation and therefore faster than they're willing to pay for it, then standards "happen". This is the existing practice and experience play in standards. Your customers are too willing – they want choice. Your competitors are too willing – it's the thin edge of the wedge. Think about componentization. Think about the LAMP stack.
Doesn't work to the left of the red line – it's too early. (Failed stds are attempted too early.)

Of course it's more complex than this …



Animations: 1. Start with the back axis and minicomputer curve. 2. Add UNIX. 3. Add Linux. 4. Add the front plane axis and the PC curve.

Notes: See next slide!

Notes for previous slide:
UNIX was about taking out DEC's dominance in minicomputers. UNIX in the late 80s was less secure, less robust, less scalable than any VMS system. It didn't stop us from buying it (especially in the face of DEC's underhanded treatment on 3rd party memory and disks). And think about this in terms of Microsoft's emphasis on TCO, security, maturity, etc.

PCs happened in a different plain of competition. They competed with non-consumption, i.e. UNIX didn't loose the desktop to the PC.

Example from my history: 1985 we told the user we could deliver the latest system with 3 people over 2 years and we could start in a year. Then asked her about the PC on her credenza that she had hidden in her budget and with which she was modeling data.

PC evolved to the point of being interesting as departmental servers (sort of a next generation minicomputer), and then along came linux as a better replacement for those expensive UNIX boxes.

Prior to the point where they begin to over deliver, the market leader is often offering the technology in a tightly integrated fashion and best delivers to consumer needs in this space where the solutions typically are not yet good enough

Christensen's Next Great Observation: Value moves to adjacent nodes in the network when the node starts to commoditize

Applying the Open Source and Standards Business Tools

To buy vs build we add borrow and share

Notes: Brief intro to where we're going.
Think IBM and Apache and Linux
Think SAP and SAPDB
Think Sun and the Gnome desktop

Companies are economically rationale in their community participation

Community participation is a market conversation with your customer

Notes: The Cluetrain Manifesto by Doc Searles et al. Thesis #1: Markets are conversations. When IBM joined the Apache community we all laughed. And then we saw their “real” colours when they sold Websphere. But IBM plays by the rules. Every Apache user is potentially a Websphere customer. WiX has been a huge conversation with customers for Microsoft. It’s blogging for programmers.

Sponsor the community and you “anchor” the conversation

Notes: It’s not about control. It’s about influence.
It’s about setting the rules for the community.
It’s about setting the rules for the conversation.
Think Sun and the JCP. Think IBM and Eclipse
I’ve cynically observed in the past that this is Sun pretending to do a standard and IBM pretending to do an OSS project but the reality is despite the controls in place they are having appropriate discussions with their (and each others’) respective customers.

My Favourite Big Company Examples

Notes: Essentially walk through my favourite examples.
IBM and Apache (develop a complement quickly), Linux (manage the AIX curve, essentially change the slope of Moore's curve).
SAP and 100 people times 2 years to develop SAPDB then publish it for free under the GPL: expand into the middle market by giving away the complement to avoid the database tax, while salting the fields with the GPL, and then hand off the community management to MySQL AB.
Sun acquired and developed a lot of accessibility technology, provided the professional fit and finish to add it to the Gnome desktop, and got an entire desktop in return.

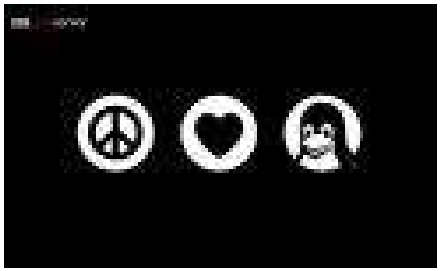
The Microsoft Slide

Notes: Talk about WiX, and its success. Talk about the opportunity for Microsoft with source other than core product code. Talk about some of the war stories. Talk about the positioning of Shared Source.

Think of OSS as “just another tool” and community as “just another customer engagement” mechanism and it can become a better way to serve customers

Notes: Point out I am being deliberately off hand in this statement here.
OSS is a great way to start new businesses – whether you’re building complements to a core offering, or starting an upstart disruptive business model from “cheap off the shelf parts”. OSS is an aggressive way to publish and salt the fields around you while making customers happy and having a conversation with them. Drive your competitors nuts. It is also NOT a business panacea.

If we don't apply "business economic" models to OSS we fall victim to the continued "Summer of Love", Anti-establishment, "communist manifesto" political labeling of those businesses most threatened



Notes: If we don't get this right, the brands at the top of the screen get labeled by the brands at the bottom of the screen.

Understanding the context of OSS in business will allow us to grow faster using well understood business tools practices

Commerce and Politics exists within the State: THAT economic debate has been going on since we had government

Notes: Unfortunately, as long as we look like a political movement that thinks it's special, we will get boxed that way.

Seeing OSS as a business tool as well as a “movement” rebalances the conversation

Questions?

stephe@optaros.com
http://stephesblog.blogs.com