What I’m Going to Talk About

Business models for an industry shaped by open source.

NOT

Business models for “open source businesses.”
A Pattern Language

“Patterns are all the brain knows about.”

- Jeff Hawkins

A Memory-Prediction Engine
But What if the Patterns Are Wrong?

“The map is not the territory.”
--Alfred Korzybski

Pattern Failure In The PC Revolution

• 1981: IBM PC built of commodity components

• Market expands a million-fold, breaking IBM’s industry dominance

• Dell becomes #1 by embracing commodity economics; IBM eventually abandons market

• Software separate from hardware: IBM signs away future to Microsoft
Desktop Application Stack

- Proprietary Software (Control by API)
- System Assembled from Standardized Commodity Components
- Hardware Lock In By a Single-Source Supplier

Pattern Failure at Work!

- How many of you use Linux?
- How many of you use Google?
- Where are the user-friendly applications on Linux?
User Friendly Applications on GNU/Linux (or BSD)

Google

amazon.com.

PayPal®

Yahoo! GetLocal

Another Pattern Failure?

These applications are created by open source developers and run on an open source platform, but…

– Companies are fiercely proprietary
– Source code is not distributed
– Licenses have no effect
– Value is in data, not just code
"The Law of Conservation of Attractive Profits"

"When attractive profits disappear at one stage in the value chain because a product becomes modular and commoditized, the opportunity to earn attractive profits with proprietary products will usually emerge at an adjacent stage."

-- Clayton Christensen
Author of The Innovator's Solution
In Harvard Business Review, February 2004

Desktop Application Stack

Proprietary Software (Control by API)
System Assembled from Standardized Commodity Components
Hardware Lock In By a Single-Source Supplier
What Is a Pattern?

“Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.” (Christopher Alexander)

– Context: Previously High-Value Industry Being Commoditized
– Problem: Build or maintain value
– Solutions: The subject of this talk
Three Business Model Contexts

• Integration of commodity components
• Internet application services “up the stack”
• Subsystem-level lock-in

Integration and Assembly
Format of Design Patterns

• Brief statement of situation or problem.

• Examples and discussion that demonstrate the generality of the situation.

• A prescription that expresses the solution.

Design for Participation

An open source operating system consists of “small pieces loosely joined.”

Discussion: Linux, Apache, Perl and CPAN

Therefore: Architect your software in such a way that it can be used easily as a component of a larger system. Use a license that does not hinder such a combination. Keep your software modular, and make certain that you document all of the interfaces.
User-Centered Development

The benefits of open source come from sharing your development efforts and processes with your users.

Therefore: Release early and release often. Set up mechanisms for users to submit bugs and patches. Promote your most active users into roles of greater responsibility.

Don’t Differentiate on Features

When useful components are abundant and free, adding proprietary features will only slow you down.

Therefore: Focus your development efforts on speed of testing, assembly, and integration so that your users can always have the best components that the marketplace has to offer.
Follow Industry Standards

Users want choice, but not too much of it.

Therefore: Offer your products in a number of proven, industry-standard configurations, and let your users choose between them. As new application domains emerge, develop configurations to support them.

Software as a Service

Users don’t want computers, they want the services they provide.
The Perpetual Beta

When devices and programs are connected to the internet, applications are no longer software artifacts, they are ongoing services.

Therefore: Do not package up new features into monolithic releases, but instead add them on a regular basis as part of the normal user experience. Engage your users as real-time testers, and instrument the service so that you know how people use the new features.

Leverage Commodity Economics

Scale matters. A successful internet service may need to scale up rapidly, at low cost.

Therefore: Use Linux, Apache, and other open source components running on commodity PC hardware as the basis for any internet service. Arrange these components in fault-tolerant arrays, with management tools that minimize the number of required sysadmins.
Users Add Value

The key to competitive advantage in internet applications is the extent to which users add their own data to that which you provide.
Users Add Value

The key to competitive advantage in internet applications is the extent to which users add their own data to that which you provide.

Therefore: Don’t restrict your “architecture of participation” to software development. Involve your users both implicitly and explicitly in adding value to your application.
Network Effects by Default

Only a small percentage of users will go to the trouble of adding value to your application.

Therefore: Set inclusive defaults for aggregating user data as a side-effect of their use of the application.

The Long Tail

Many limiting factors from the physical world are absent from the internet.

Source: Wired Magazine
The Long Tail

Many limiting factors from the physical world are absent from the internet.

Therefore: Use the power of the computer to monetize niches that formerly were too small to be commercial.

The Mechanical Turk

Services need to be robust, but because they are typically data-driven, they also need to be updated constantly.

Therefore: Build your core services infrastructure with traditional software components written in languages like Java, C, and C++, but build your interfaces with templating systems and with dynamic languages like Perl, Python, and PHP. Think of your programmers, designers, and admins as part of the application.
Von Kempelen’s Mechanical Turk

Software Above the Level of a Single Device

The PC is no longer the only access device for internet applications, and applications that are limited to a single device are less valuable than those that are connected.

Therefore: Design your application from the get-go to integrate services across handheld devices, PCs, and internet servers.
Platforms and Tools (The Eclipse Pattern)

Third party developers are an essential part of the success of new platforms.

*Therefore:* Reduce complexity for developers by building tools that specifically support new classes of application, and use those tools to simplify access to underlying components and services.

Component-Level Lock-In
**Intel Inside**

Inside every open system there are proprietary, single-source components. The open PC architecture has Intel Inside; the open internet has Cisco Inside.

*Therefore:* Look for opportunities to be the single source for an essential subsystem of an open system.

**Dual Licensing**

The GPL allows special rights to the creator of the software to distribute the software under other terms.

*Therefore:* Make sure to obtain copyright assignment for all contributed code.
Data is the Next “Intel Inside”

Applications are increasingly data-driven.

Therefore: Owning a unique, hard-to-recreate source of data may lead to an Intel-style single-source competitive advantage.
Welcome to NAVTEQ

We are a world leader in premium-quality digital map data. You'll find NAVTEQ data onboard most in-vehicle navigation systems sold in North America and Europe. It's also the driving force behind a new generation of important navigation services, including Internet websites, Enterprise/Fleet/GIS solutions and Location Based Services (LBS).

NavTeq Since IPO
Own the Namespace

Some data is a commodity and impossible for any one party to own, but access to the data can be controlled through legal means.

*Therefore*: If you can’t own the data, use legal means to own the namespace or registry for the data.

“I’m an inventor. I became interested in long term trends because an invention has to make sense in the world in which it is finished, not the world in which it is started.”

-Ray Kurzweil
For more information

http://www.oreilly.com
http://tim.oreilly.com
http://conferences.oreilly.com
http://www.oreillynet.com