Technology and Strategy

Modularity, open interfaces, open standards and open source
Michael A M Davies
Strategy and technology

Demand opportunity
- segments
- behaviour change
- diffusion and adoption
- chasm

Technological infrastructure
- architecture
- parameters
- envelope and trade-offs
- innovation trajectory

Co-evolution & transitions
- episode, era
- dominant design
- radical vs incremental
- architectural vs modular
- disruptive

Business ecosystems, value creation and value capture
- niche
- lead/follow
- co-opetition
- inimitability
- focus on locus of value
- standards
- modularity

Decision making
- common goals, balanced power
- focus on timely, objective facts
- embrace uncertainty
- several options

Michael A M Davies
Two simultaneous challenges: create value, at the same time ensure that you can capture value.
Some terminology for timelines and transitions

*Epoch* applies to the beginning of a new period marked by radical changes and new developments - while *era* applies to the entire period.

Roget's New Millennium™ Thesaurus, First Edition (v 1.3.1) © 2007

Michael A M Davies
Era

noun

1. a period of time marked by distinctive character, events and so on
2. the period of time to which anything belongs or is to be assigned
3. a period of time as reckoned from a specific date serving as the basis of its chronological system

Epoch

noun

1. a period of time marked by distinctive character, events and so on\(^1\)
2. the **beginning** of a distinctive period in the history of anything\(^1\)
3. a **point of time** distinguished by a particular event or state of affairs\(^1\)
4. a notable event that marks the **beginning** of a period of history, especially one considered remarkable or noteworthy\(^2\)

---

\(^1\): Random House Unabridged Dictionary, © Random House Inc. 2006
\(^2\): American Heritage® Dictionary, © 2000 Houghton Mifflin
Episode

*noun*

1. a portion of a narrative that relates an event or a series of connected events and forms a coherent story in and of itself\(^1\)
2. an incident in the course of a series of events\(^2\)
There are distinct stages in the battle for technological dominance

<table>
<thead>
<tr>
<th>Phase I</th>
<th>Phase II</th>
<th>Phase III</th>
<th>Phase IV</th>
<th>Phase V</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₀</td>
<td>Tₚ</td>
<td>Tₗ</td>
<td>Tₕ</td>
<td>T₅</td>
</tr>
</tbody>
</table>

- **R&D Build Up**: Beginning of a new technological field, with an organization pioneering applied R&D.
- **Technical Feasibility**: First working prototype emerges.
- **Creating the Market**: First launching of a commercial product.
- **Decisive Battle**: Clear early front runner appears.
- **Post-Dominance**: One of the early designs becomes dominant.


Massachusetts Institute of Technology
These phenomena often result in characteristic {product | industry} ‘life-cycle’ (YMMV)

<table>
<thead>
<tr>
<th>Demand Opportunity</th>
<th>Business Ecosystem</th>
<th>Technological Infrastructure</th>
<th>Maturity</th>
<th>Eclipse or renewal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early ferment</td>
<td>Dominant design emerges</td>
<td>Incremental innovation</td>
<td>Maturity</td>
<td>Eclipse or renewal</td>
</tr>
<tr>
<td>Lead users, early adopters - high payoff, low switching costs</td>
<td>Early mainstream - usability, cost more important</td>
<td>Mainstream customers - soft factors, aesthetics</td>
<td>Saturation, segmentation, customization</td>
<td>Find new needs or new customers</td>
</tr>
<tr>
<td>Many entrants - diverse business models</td>
<td>Decisive battles for leadership</td>
<td>Intensifying competition, early consolidation</td>
<td>Fierce competition, consolidation around majors and minors</td>
<td>Select optimal architecture, drive down costs, focus on ease of use</td>
</tr>
<tr>
<td>Make it work - innovate on performance, diverse integrative designs</td>
<td>Select optimal architecture, drive down costs, focus on ease of use</td>
<td>Provide broader offer, rationalize portfolio, build complementary assets</td>
<td>Develop broad portfolio, build platforms</td>
<td>Search for new options</td>
</tr>
</tbody>
</table>

Michael A M Davies
Reconciling these two models

<table>
<thead>
<tr>
<th>Time</th>
<th>Demand</th>
<th>Early fermentation</th>
<th>Dominant design emerges</th>
<th>Incremental innovation</th>
<th>Maturity</th>
<th>Eclipse or renewal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase II</td>
<td>Follow the leaders</td>
<td>Lead users, early adopters</td>
<td>Early</td>
<td>Mainstream</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase III</td>
<td>Lead users, early adopters</td>
<td>Early</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase IV</td>
<td>Early</td>
<td>Mainstream</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase V</td>
<td>Late majority</td>
<td>Maturity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Technical Feasibility | Creating the Market | Decisive Battle | Post-Dominance

Infrastructure | diverse integrative designs | costs, focus on ease of use | complementary assets | build platforms

Michael A M Davies

GNU and Linux

• (Lead) user (from CSAIL at MIT) wanted **control**
• Developed key complements:
  – compiler and editor
  – license – GPL/“copyleft”
• Linus Torvalds: ‘91, <10k lines, Intel’s ‘386 architecture
• Increased functionality
• Wider variety of hardware platforms
• Network effects

• Increasing complexity
Red Hat

• (Lead) user finds Linux difficult to install and use
  – couldn’t afford a Unix workstation
  – The Halloween Release
• Initially inferior, did not appeal to mainstream, potential for rapid innovation, surpasses performance
• Combine build with distribution channel
• Key innovations
  – installer
  – package manager
• Time-paced release – work back from ship date
Red Hat - update

• 2004 ranks first in value amongst CIOs
• Acquires AOL’s Netscape server software
• Partners with IBM
• 2005, Enterprise Linux 4 wins OS and Server Product of the year at Techworld
• IPO in August 1999 – shares triple

• ~2,200 employees
• ~$500 million in revenue, ~$4,000 million market cap
Microsoft’s take on Linux

- OSS poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space.
- The intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.
- Commercial quality can be achieved / exceeded by OSS projects.
- OSS is long-term credible FUD tactics can not... combat it.
- Linux outperforms many other UNIXes.
- Linux can win as long as services / protocols are commodities.
- OSS projects have been able to gain a foothold in many server applications because of the wide utility of highly commoditized, simple protocols. By extending these protocols and developing new protocols, we can deny OSS projects entry into the market.
- OSS evangelization scales with the size of the Internet much faster than our own evangelization efforts appear to scale.
Why does open source work (economically, not technically)?

<table>
<thead>
<tr>
<th>Co-operate</th>
<th>Compete</th>
</tr>
</thead>
<tbody>
<tr>
<td>⇒ create value</td>
<td>⇒ capture value</td>
</tr>
<tr>
<td>• OS infrastructure, not a basis for differentiation</td>
<td>• Deliver <strong>complements</strong></td>
</tr>
<tr>
<td>• “...all the users of that infrastructure have a reason to help you build it and ensure that it continues to develop.”</td>
<td>• Package sources and patches</td>
</tr>
<tr>
<td>• A modular system – enabling decoupled development</td>
<td>• Provide professional services</td>
</tr>
<tr>
<td>• Expose interfaces and inner workings of modules</td>
<td>• Sell hardware that runs Linux</td>
</tr>
</tbody>
</table>

Michael A M Davies
Modularity is becoming more prevalent, increasing value *creation*, challenging value *capture*

- Falling costs of co-ordination make modularity easier
- Modularity, where it can be employed effectively, can accelerate value creation
  - once dominant design established, hence stable architecture and modular interfaces
  - and ultimate performance is not critical
  - autonomous or modular innovations, in this context
- Very challenging for value capture
  - loss of control for leaders
  - rapid, diverse innovation
  - revenues and value widely dispersed
Modularity

noun

1. the use of individually distinct functional units, as in assembling an electronic or mechanical system\(^1\)
2. designed with standardized units or dimensions, as for easy assembly and repair or flexible arrangement and use\(^2\)


Modularity $\rightarrow$ decoupling

• “When a product or process is ‘modularized,’ the elements of its design are split up and assigned to modules according to a formal architecture or plan.”

• “From an engineering perspective, a modularization generally has three purposes:
  – to make complexity manageable
  – to enable parallel work
  – to accommodate future uncertainty”

## Interfaces

<table>
<thead>
<tr>
<th>Modular</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Customer understands and can specify key parameters</td>
</tr>
<tr>
<td>• Can be measured and tested reliably and unambiguously</td>
</tr>
<tr>
<td>• Understand how variation affects system performance</td>
</tr>
<tr>
<td>• Market can function effectively</td>
</tr>
<tr>
<td>• Codified knowledge</td>
</tr>
<tr>
<td>• Difficult to protect</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interdependent/ Systemic/Integral</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Associated with optimizing design for ultimate performance</td>
</tr>
<tr>
<td>• Unstructured technical dialogue</td>
</tr>
<tr>
<td>• Necessary information for market does not exist</td>
</tr>
<tr>
<td>• Management and integration most efficient coordinating mechanisms</td>
</tr>
</tbody>
</table>

Michael A M Davies
IBM System/360

- First modular computer, conceived as a family of computers
  - different sizes suitable for different applications
  - same instruction set
  - standard interfaces for peripherals
- Design rules and decentralized development
  - Central Processor Control Office defines rules
  - each team full control over hidden elements
- Wildly successful, drove other players out of the market
- BUT undermined IBM’s dominance in the long run - through emergence of plug-compatible modules
There are six modular operators that together enable a very wide range of system designs:

- **Splitting**: Separating systems into modules that interact across well-defined interfaces.
- **Substituting**: Switching between components that perform the same function.
- **Augmenting**: Adding a module to increase the functions of the system.
- **Excluding**: Removing a module to reduce the functions the system can perform.
- **Inverting**: Making an embedded function into a stand-alone module.
- **Porting**: Moving a module from one system to another.
The resulting systems can exhibit several different types of modularity:

- **Slot**
- **Direct**
- **Bus**

**Component Swapping**

**Component Sharing**

**Bus**
Modular platforms can be a very effective vehicle for diverse offers.
### Strategic options

<table>
<thead>
<tr>
<th>Architect</th>
<th>Module player</th>
</tr>
</thead>
<tbody>
<tr>
<td>• For system innovations, may require broad scope of activities at the outset</td>
<td></td>
</tr>
<tr>
<td>• Create design rules, define visible information</td>
<td></td>
</tr>
<tr>
<td>• Convince people this architecture will prevail</td>
<td></td>
</tr>
<tr>
<td>• As modularity established, lead the evolution of the business ecosystem</td>
<td></td>
</tr>
<tr>
<td>• Conform to the architecture, interfaces and test protocols established by others</td>
<td></td>
</tr>
<tr>
<td>• Master the hidden information involved</td>
<td></td>
</tr>
<tr>
<td>• Rely on superior execution</td>
<td></td>
</tr>
</tbody>
</table>
Linux 0.01

The architecture of Linux is significantly more modular than the architecture of Mozilla
And than OpenSolaris
…or XNU

Mozilla – before and after a purposeful re-design effort – modularity to allow participation

The payoff...

Longitudinal Evolution of Mozilla's Propagation Cost

- Propagation Cost
- Size

Open source code has much lower propagation cost than conventional proprietary approaches.

<table>
<thead>
<tr>
<th>Product Category</th>
<th>“Open”</th>
<th>“Closed”</th>
<th>Test Stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Financial Mgmt</td>
<td>7.74%</td>
<td>56.06%</td>
<td>p&lt;0.1%</td>
</tr>
<tr>
<td>2: Word Processing</td>
<td>8.25%</td>
<td>41.77%</td>
<td>p&lt;0.1%</td>
</tr>
<tr>
<td>3: Spreadsheet</td>
<td>23.62%</td>
<td>54.31%</td>
<td>p&lt;0.1%</td>
</tr>
<tr>
<td>4a: Operating System</td>
<td>7.18%</td>
<td>22.59%</td>
<td>p&lt;0.1%</td>
</tr>
<tr>
<td>4b: Operating System</td>
<td>7.21%</td>
<td>24.83%</td>
<td>p&lt;0.1%</td>
</tr>
<tr>
<td>5: Database</td>
<td>11.30%</td>
<td>43.23%</td>
<td>p&lt;0.1%</td>
</tr>
</tbody>
</table>
Key to value capture is focus on locus of value

- “... the virtuous virtuals have carefully nurtured and guarded the internal capabilities that provide the essential underpinnings of competitive advantage... they invest considerable resources to maintain and extend their core competences [because without them] their strategic position in the network would be short-lived”
  

- “Attractive profitability seems to flow ... to the point at which unsatisfied demand for functionality, and therefore technological interdependency exists.”
  

Michael A M Davies
Locus

noun

1. a center or focus of great activity or intense concentration
2. a center or source, as of activities or power
Amdahl’s Law: “…make the common case fast…”

- Amdahl’s Law is concerned with the speedup achievable
  - from an improvement to a computation
  - affects a proportion $P$ of that computation
  - where the improvement has a speedup of $S$

- Amdahl's Law states that the overall speedup of applying the improvement will be

$$\frac{1}{(1 - P) + \frac{P}{S}}$$

“God grant me the serenity to accept the things I cannot change (much); courage to change the things I can (a lot); and wisdom to know the difference.”

- Reinhold Niebuhr
Sun Microsystems – intense focus on performance bottleneck

Image removed due to copyright restrictions.
Interestingly, Sun Microsystems has become a major open source advocate

- Made Java™ open source
- OpenSolaris
  - build a developer community
  - derived from Unix System V – only open version
- OpenSPARC
  - processor architecture design, application development
  - building community around it to advance it
- OpenStorage
  - built around ZFS
  - combine open source software with industry standard hardware
Leveraging open source requires a disciplined approach, from legal and strategic perspectives.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-house development</td>
<td>Practical and pragmatic procedures for evaluating open source options</td>
</tr>
<tr>
<td><strong>Open</strong></td>
<td><strong>Closed</strong></td>
</tr>
<tr>
<td>In-house development</td>
<td>Develop own software, and release to open source community</td>
</tr>
<tr>
<td>Drive how to access</td>
<td>Develop own software and keep source and rights internal</td>
</tr>
<tr>
<td>software</td>
<td></td>
</tr>
<tr>
<td>Custom order</td>
<td>Encourage new open source community development [Google with Android]</td>
</tr>
<tr>
<td>Existing 3rd party</td>
<td>Pay for development as “work for hire” and keep rights internal</td>
</tr>
<tr>
<td>Integrate 3rd party open source components</td>
<td>Buy rights to use off-the-shelf 3rd party software</td>
</tr>
</tbody>
</table>

Source: Olswang Open Source Summit, 9 Nov 2007; private discussion and communication with Nigel Swycher of Olswang, and Heather Meeker of Greeberg Taurig, and others

Michael A M Davies
The key concern with open source is inherited obligations, “copyleft” licenses

<table>
<thead>
<tr>
<th>Principal form of License</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Public License – GPL of the Free Software Foundation</td>
<td>“Copyleft” – most open; prohibits proprietary forks, strongly driving openness</td>
</tr>
<tr>
<td></td>
<td>If you add code, and distribute software, you must grant unaltered GPL to combined program - Linux Kernel uses this (GPLv2)</td>
</tr>
<tr>
<td>Apache Software License (ASL)</td>
<td>Allows proprietary extensions or distributions</td>
</tr>
<tr>
<td></td>
<td>Code issued with this license can be included on closed products</td>
</tr>
<tr>
<td></td>
<td>Much of Android software uses this (Apache v2)</td>
</tr>
<tr>
<td>Berkeley Software Distribution (BSD Unix)</td>
<td>Few restrictions on what you can add and how you can limit openness of what you add</td>
</tr>
<tr>
<td></td>
<td>Apple uses this</td>
</tr>
</tbody>
</table>

Michael A M Davies
Resources

- http://opensource.mit.edu
- http://www.opensourcestrategies.org/
- http://www.osxbook.com/
- http://www.kernelthread.com/mac/osx/
- http://arstechnica.com/reviews/os/mac-os-x-10-5.ars
- http://www.roughlydrafted.com*
- http://opensolaris.org/os/