Integrated Development Environments (IDEs)

Technology Strategy

Chad Heaton
Alice Park
Charles Zedlewski
# Table of Contents

Market Segmentation.............................................................................................................. 4  
When Does the IDE Market Tip? ........................................................................................... 6  
Microsoft & IDEs ................................................................................................................... 7  
Where is MSFT vulnerable? ................................................................................................. 11  
Eclipse & Making Money in Open Source ........................................................................ 12  
Eclipse and the Free Rider Problem ..................................................................................... 20  
Making Money in an Eclipse World? .................................................................................... 14  
Eclipse vs. Microsoft: Handicapping the Current IDE Environment .................................. 16  
Requirements for Eclipse success.......................................................................................... 18
Overview of the Integrated Development Environment (IDE) Market

An Integrated Development Environment (IDE) is a programming environment typically consisting of a code editor, a compiler, a debugger, and a graphical user interface (GUI) builder. The IDE may be a standalone application or may be included as part of one or more existing and compatible applications. Major innovation cycles over the 30-year history of IDEs have generally addressed improvements in performance as measured by the product’s ability to increase the productivity of programmers who use such tools (e.g. lines of code written per hour).

Regression of Lines of Source Code vs. Machine Code

Thus, value created is relatively easy to measure. However, value capture is a more complex issue as, historically, firms within this space have taken different approaches to generating profits.

Historical Industry Dynamics

In general, firms compete via the three major strategic approaches summarized below:

Summary of Traditional Value Creation / Value Capture Models for IDE’s

---

<table>
<thead>
<tr>
<th>Model</th>
<th>Type of Firm</th>
<th>Pricing</th>
<th>Product Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniqueness Play</td>
<td>Pure play IDE vendor</td>
<td>• License $$/developer seat ($100’s to $1,000’s)</td>
<td>• Compatible with multiple platforms</td>
</tr>
<tr>
<td></td>
<td>Ex: Borland, WebGain</td>
<td>• + ann. maintenance fee</td>
<td>• Differentiates thru high functionality/flexibility</td>
</tr>
<tr>
<td>Complementary</td>
<td>Platform Software Firms</td>
<td>• Significant discount or FREE</td>
<td>• Usually incompatible with other platforms</td>
</tr>
<tr>
<td>Assets Play</td>
<td>Ex: IBM, BEA, SAP</td>
<td>• High margins on other products &amp; services</td>
<td>• Differentiates thru value of complementary product stack</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• IDE = “hook-in” to product family; create lock-in</td>
</tr>
<tr>
<td>The “Hybrid” Play</td>
<td>Unique to Microsoft</td>
<td>• Sizable margin on IDE license</td>
<td>• Incompatible to other platforms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• + ann. maintenance fee</td>
<td>• Differentiates thru high performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• + high margins on other products &amp; services</td>
<td>• IDE creates lock-in to “Microsoft Shop” of products</td>
</tr>
</tbody>
</table>

The Uniqueness Play is extremely difficult to sustain since low or no cost substitutes are available and extremely large, deep-pocket software platform companies (such as Microsoft) dominate the market. Borland is one of the few surviving firms able to maintain a leadership market position using this strategy. Instead, the dominant IDE strategy has been to sell IDEs at a significant discount (or give it away for “free”) in order to stimulate demand for other revenue generating products. For platform product vendors (e.g. BEA, Oracle, IBM), IDEs help grow their platform install base such that customers will become increasingly locked-into their technology stacks and have high switching costs to other platforms. Incompatibility between platforms is an attempt to create even stronger lock-in forces. Microsoft is something of an exception. The company captures some value directly from its IDE but captures still more from the IDE’s linkages to its platforms.

**New Entrant**

Eclipse is an open source IDE project launched in 2000. Eclipse represents the first new “competitor” in the IDE market in several years. Eclipse’s product and business model is unique and promises to change the strategies of other vendors and complementors.

**Market Segmentation**

In general, IDEs are relatively homogenous since they are tools that build other applications and...
thus must remain extremely versatile and non-specific. IDE users generally fall into one of four groups.

<table>
<thead>
<tr>
<th>Corporate Software Developers</th>
<th>Low Developer Sophistication</th>
<th>High Developer Sophistication</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Department Developer</td>
<td>• Makes $40-60k, 1-5 years experience</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Does simple “tactical” development projects (e.g. a data import script)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Writes code in simple languages like Visual Basic, Perl, html, etc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Decentralized</td>
<td></td>
</tr>
<tr>
<td>The Corporate IT Developer</td>
<td>• Makes $70-100k, 3-10 years experience</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Does central “strategic” development projects (e.g. a new CRM system)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Writes code in OO languages like Java, C#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Part of a centralized corporate organization</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Independent Software Vendor Developers</th>
<th>The Mom &amp; Pop Shop</th>
<th>The Enterprise ISV</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Develops small, practical software applications for the desktop or for narrow enterprise niches</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Bootstrapped</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Limited need for defined product architecture, API’s, documentation, integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Develops high ticket enterprise software infrastructure and applications for large corporations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Often venture backed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Products architected for scale, API’s, integration with 3rd party apps</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each sub-market represents a set of customers with varying needs and levels of sophistication that determines the most influential purchase decision criteria:

1. The Department Developer is generally a low- to medium-skilled programmer who builds relatively simple applications/tools for a specific function that need not be integrated into the corporate technology infrastructure. These developers need tools that enable straightforward and productive development in widely used languages such that they are easily transferable. Easily implemented complementary IDE plug-in tools are also essential in order to meet evolving needs and pressures to develop at decreased time-to-market. Thus, these developers will look for an economical, off-the-shelf IDE product that is easy to use and widely supported. Vendors who target this market must have strong distribution channels and focus on providing an extremely high performance product. Microsoft is dominant in this segment for these reasons and is often differentiated by its superior service network and training.

2. The Mom & Pop Shops generally have low developer sophistication and work in decentralized
environments. As such, they seek widely used products with basic functionality. However, because this segment is much more concerned with price, one would think the low cost alternative (such as the Eclipse IDE) may have a competitive advantage. Instead, Microsoft is extremely successful in this segment due to its superior retail distribution channel and large, trained developer community.

(3) The Corporate IT Developer is considerably more sophisticated and develops strategic applications within a centralized and complex technology infrastructure. This segment is interested in software products that exhibit flexibility to customize and interoperate with the existing (usually heterogeneous) enterprise platform. This is an extremely attractive market for vendors as corporations are willing to spend top dollar for the most advanced products. Corporate CIO’s are often concerned with vendor lock-in and prefer to have flexibility in choosing among several software providers. Thus, in this market segment, Java-based IDE vendors (e.g. Borland) and ISVs have an advantage since they support multiple platforms and reduce customer lock-in.

(4) The Enterprise ISVs are also extremely sophisticated and seek IDEs that most fully support their desired performance and feature requirements. Since their requirements tend to be highly specific, complementary IDE add-ons will be relatively less valuable. Second, Enterprise ISVs value multi-platform support capabilities because they effectively increase their potential client base across multiple platform architectures.

When Does the IDE Market Tip?

The IDE market has threshold network effects. An IDE is more valuable to its customers when there are already 1 million developers using the IDE. In the same way, the IDE does not have complete network effects, as the IDE is not that much more valuable to customers when there are 2 million developers using the IDE versus 1 million. Consequently in its 40 years of history, the IDE market has shown a resistance to tipping.

The relative value to the customer of a large IDE install base varies from segment to segment. Large organizations (Enterprise ISV, Corporate IT) need a critical mass of developers trained on an IDE,
but not mass adoption. If the product is sufficiently unique, both customer types can invest in getting the appropriate trained developers on board. In contrast, small organizations (Department Developer and Mom & Pop) don’t have the resources to keep a very diverse developer group on staff. These segments are more sensitive to an IDE’s install base, preferring to stick with “mainstream” technology. These customer preferences shape the propensity of each segment to tip.

**Marketing Tipping Thresholds**

![Graph showing tipping thresholds for corporate and department developers](image)

**Microsoft & IDEs**

Microsoft was the originator of the IDE platform business model for value creation and capture that so many software companies emulate today. As a dominant vendor, it is worth examining the origins of the Microsoft franchise and where it stands today to better understand the competitive dynamics of the industry in its current form.

**Brief History of MSFT and IDEs**

Microsoft was founded in 1975 as a programming tools company that sold interpreters for the BASIC programming language. Microsoft’s next two programming language products were Fortran and COBOL compilers released in 1977 & 1978 respectively.

These early interpreters and development tools were little more than text editors with built-in compilers. They lacked the visual programming elements that comprise today’s IDEs. While there are
different theories regarding the subsequent history, most agree that Bill Atkinson created the first visual
development environment known as Hypercard. Hypercard was shown to Microsoft, who then began to
emulate its visual development capabilities in their own IDE. The result was the first version of Visual
Basic released in 1991. Subsequent releases occurred every 1-2 years through the most recent version,
Visual Studio.NET 2005 (currently in Beta).

Strategic Importance of IDEs to MSFT

Microsoft has most completely exploited the linkages between the IDE and other complementary
software products. When an organization chooses to develop software with Visual Studio, it biases the
organization’s decision to standardize on Microsoft’s operating systems, web server, database and mobile
platforms. Visual Studio also has tight linkages to Microsoft proprietary standards for technologies like
user interface design and web services. Thus a purchase of a Visual Studio product at $500 per developer
can potentially influence purchase decisions for thousands of dollars worth of Microsoft software.

The interrelationship between the IDE and the platform can be represented as a series of positive
feedback loops.
By investing to make Microsoft IDEs the simplest, most productive tools in the marketplace, they encourage developers to write Microsoft platform-based applications. The proliferation of the Microsoft platform thereby encourages developers to write Microsoft applications as they wish to target the largest possible installed base, growing Microsoft’s IDE share.

We have seen that Microsoft captures value not directly from the IDE but through complementary product sales and that this approach to value capture represents a positive loop. Microsoft further reinforces its position in the IDE market with an array of complementary assets noted below.
Many of these complementary assets fit the classic definition of a tightly held firm resource: difficult to observe, unable to acquire on the open market or develop in a compressed timeframe.

Competitors to Microsoft like IBM, Borland, Oracle and BEA can develop products as compelling as Visual Studio, but are challenged to replicate Microsoft’s complementary assets.

**Visual Studio as Sword and a Shield**

Visual Studio has also acted as a bridge which lends strength from Microsoft platforms which are already entrenched (e.g. Windows) to new Microsoft platforms which are comparatively weak (e.g. Windows Mobile, distributed computing).

Web Services (and their underlying XML architectures) represent a new, platform-agnostic method for computers to share data over the public internet. For instance, using Web Services, one Windows-based system could seamlessly transmit customer order information to another Unix-based system that ultimately transforms and forwards the data to a mainframe. How does Microsoft promote the Windows/SQL Server platform under these conditions? Much of the answer lies with its IDE. Microsoft invested in Visual Studio.NET to make it the simplest, most rapid tool for designing and implementing Web Services. In fact, with only one line of code, any function written with Visual Studio.NET can be transformed into a Web Service. All of the underlying XML “plumbing” code is generated automatically. Thus, even though Web Services could be written in any language for any
platform, Microsoft used its significant cash position to ensure that they would be easiest to write and implement on the Windows platform.

One sees a similar approach with mobile devices. In the latest incarnations of Visual Studio.NET, Microsoft has embedded functionality that will automatically detect and adapt for the device type that the code is being executed on. Thus, a developer who chooses to write his/her application using Visual Studio will automatically have the capability to run that application on PC’s, cell phones, PDA’s, with a particular preference for Microsoft OS based mobile devices. Developers come to Visual Studio for the functionality and the access to ubiquitous platforms like Windows. They may leave developing mobile applications which would buttress Microsoft’s languishing platforms like Windows Mobile.

Where is MSFT vulnerable?

Despite having tremendous complementary asset and cash advantages over its competitors, Microsoft’s IDE does face some areas of vulnerability.

Heterogeneity – One area where Microsoft is clearly weak (by choice) is in its IDE support for competing platforms. That is, while Microsoft’s tools can work in a heterogeneous environment, they clearly work best when incorporated with their own proprietary software products. But the reality is that most large enterprises have computing environments consisting of some combination of Unix, Linux, Mainframe, Oracle, Apache, IBM, and Microsoft technologies. Therefore, when enterprises look to standardize their development teams on a single IDE platform, they are concerned not only with performance but also with adaptability. To the extent that Visual Studio remains a “Windows only” development environment, one can safely say the market may never completely “tip” to Microsoft’s IDE for this reason.

Lock-in Fears – Microsoft is also vulnerable to CIOs who fear vendor “lock-in”. This is particularly important for CIOs of large organizations that have considerable annual IT expenditures and very high switching costs. CIOs of such companies will place a higher emphasis on a diverse set of technology software providers to avoid these “lock-in” problems. Small technology ISVs and IT departments, on the other hand, are likely to feel relatively less threatened by “lock-in”. Thus, the relative merits of software
productivity, ease of integration, and availability of trained developers and support will be much higher for them.

**Eclipse & Making Money in Open Source**

Eclipse is a non-profit organization developing an open-source IDE supported by IBM, BEA and other large enterprise software providers. Eclipse has seen rapid uptake as judged by the several hundred thousand downloads to date. The lingering question for IDE providers sitting in Eclipse’s path (IBM, BEA, Oracle, Borland et al) is what the potential is for profits if Eclipse becomes a dominant player.

Today there are two prevailing approaches to profiting directly from open source products: “professional open source,” and “embrace and extend.”

“Professional open source” model (examples include MySQL, Sugar CRM, JBoss):

Professional open source companies develop the open standard source product almost unilaterally within their own companies. While allowing free access and contributions by third parties to open source products, over 90 percent of the engineers who create MySQL, SugarCRM and JBoss are employees of the lead contributing company. Ultimately, professional open source companies control the terms of the open source license and dictate what new project innovations occur. Professional open source companies leave room to create a “professional” edition of the product that they license under a more traditional software licensing arrangement. Historically, this model has not been successfully implemented in situations where the standard or code base is jointly owned/controlled by a number of independent vendors as is the situation with Eclipse.

“Embrace and extend” model (examples include Red Hat, Novell):

In the embrace and extend model, the core open source technology is developed by a distributed open source community with contributions from several for-profit vendors. Select embrace and extend vendors like Red Hat resell branded, copyrighted distributions of the open source product. A vendor distribution has a modest level of uniqueness and innovation beyond the open source product. Enterprise customers standardize on a particular branded distribution because of the value of simplification /
homogeneity that standardization provides. The switching costs of moving off a standard allow for some
degree of customer lock-in to a particular distribution. This lock in enables value capture for certain
vendors in the form of either a license fee to use the vendor’s copyrighted distribution (Red Hat) or
maintenance fees priced well above the marginal cost of providing the maintenance service (Novell).
Value creation and capture is possible under this model but is severely limited for two reasons:

1) The nature of the open source license makes it impossible for any one vendor to truly capture
value from significant product innovation. For example, companies that develop code to enhance
Linux are required to contribute that code back to the open source community. This prevents any
vendor from “hijacking” an open source project by adding incremental innovation to the code
base and claiming ownership over the combined entity.

2) Value capture is hampered by a limited degree of product uniqueness (due to point 1) which
limits lock-in, limits customer switching costs, that in turn limits the value capture “premium”
that vendors like Red Hat and Novell can charge.

The size and profitability of Red Hat illustrates these limitations. Linux arguably has created tens
of billions of dollars of customer value in the past five years but Red Hat is only a $125 million revenue
company with a mediocre 12% return on equity (versus 17% for the S&P500). Ten years of history and
case studies indicate value capture directly from open source is possible but only in the narrowest of
ways.

In almost all situations, open source commoditizes and depletes value capture in a particular link of
the value chain. Thus, adjacent product vendors benefit as:

1) Total value to the customer (the diameter of the pipe) increases.

2) Adjacent products’ value capture mechanisms remain intact so their percentage draw of value
flowing through the pipe stays constant.
In the case of Linux, adjacent product vendors like Dell, Oracle and IBM benefited enormously in the form of increased server, database and consulting services sales.

**Making Money in an Eclipse World?**

We know from other open source markets that it is exceedingly difficult to make money from an open source product itself. Contributions to the Eclipse code base are too widely distributed for a single vendor to take the “professional open source” approach to value capture. It might be possible for a vendor to take an “embrace and extend” model similar to that of Red Hat, particularly for small organization segments like “Mom and Pop ISV”. However, the value of a certified, copyrighted distribution of Eclipse is lower than a comparable one of Linux because the user base for Eclipse is much more technically self-sufficient than the user base of Red Hat (reducing the need for vendor handholding). Thus, a pure play vendor like Borland is out of a job.
Where money will be made in an Eclipse world is in the adjacent layers of the value chain. Various industry players are already realigning their IDE strategy to position themselves in advantageous links in the Eclipse value web.

**Eclipse Value Web**

- Opportunity to create & capture value directly
- Opportunity to create & capture value indirectly

If the Eclipse IDE remains the center of gravity for the community, various software vendors will take value creation/capture strategies that are not dissimilar from their original strategies. Companies that traditionally make money directly from development tools (e.g. Borland, IBM’s Rational Group, M7, etc) will make paid plug-ins to the Eclipse IDE with higher order functionality beyond that of a traditional IDE. Platform vendors (IBM, SAP, Oracle, BEA) will provide free or very low cost plug-ins to Eclipse that act as “hooks” that aid development on their respective platforms and technologies. This view of the Eclipse world is different from today’s view of the IDE market where platform vendors vie for a commanding share of the IDE itself in order to channel more developers to their respective platforms.

Crafty platform vendors may consider setting up alliances with complementary plug-in vendors where their plug-ins are optimized for the platform vendors’ plug-ins. This represents an opportunity to
shift the center of gravity of the Eclipse community. Eclipse would work with all platform vendor technologies but would work best with the platform that has the largest stable of optimized plug-ins.

**Eclipse Value Web – Shifting the Center of Gravity**

![Diagram of Eclipse Value Web]

---

**Eclipse vs. Microsoft: Handicapping the Current IDE Environment**

The IDE market is shaping up to be a “two-horse” race between Microsoft and the Eclipse consortium.

The following Figure illustrates how both players are positioned currently as well as the relative importance of each criterion to the four market segments.
### Handicapping the IDE Competition

<table>
<thead>
<tr>
<th>Product</th>
<th>Complementary Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performance</td>
</tr>
<tr>
<td>Comparative Advantage</td>
<td></td>
</tr>
<tr>
<td>Enterprise ISV</td>
<td>•</td>
</tr>
<tr>
<td>Mom &amp; Pop ISV</td>
<td>○</td>
</tr>
<tr>
<td>Department Developer</td>
<td>•</td>
</tr>
<tr>
<td>Corporate IT Developer</td>
<td>○</td>
</tr>
</tbody>
</table>
Requirements for Eclipse success

From our analysis of the IDE market, we know that: (1) the market for IDEs is not monolithic. Different segments have different propensities to tip and respond differently to various market strategies and (2) succeeding in the IDE market requires competitors to develop extensive complementary assets and varies by IDE segment. Given these characteristics, the following developments would be necessary in order for Eclipse to pose a formidable challenge to Microsoft’s stronghold in the IDE market:

Harmony between Java & Eclipse: Despite Eclipse’s status as one of the leading Java development platforms, Sun has held out from joining the Eclipse consortium to-date. Sun has a deep level of distrust in IBM’s intentions with respect to Eclipse. These suspicions are largely justified as Eclipse can be viewed as a way for IBM to wrest Java control from Sun (many suggest that it is no accident that IBM named it “Eclipse” – a direct attack on Sun’s brand). Ultimately, Eclipse and Sun will need to cooperate in order to prevent a Microsoft-dominated software development world. Since Microsoft controls its own programming languages and IDE, the result is a highly integrated and coordinated product. The Eclipse IDE and Java will need to have similar coordination and integration to successfully compete.

Seamless interoperability with complementary software platforms. Development efforts in the Enterprise ISV and Corporate IT Developer segments build on top of enterprise class databases, operating systems and application servers from companies like Oracle, IBM and BEA. The Eclipse IDE will need to easily integrate these disparate software applications as Microsoft’s IDE currently integrates its own stack.

Partner programs. Enterprise ISVs often rely on larger technology companies for go-to-market support. Large tech companies rely on ISVs to create demand for their products. Partner programs provide ISV’s with sales and marketing support in exchange for developing with partner IDEs on top of partner platforms. Microsoft is advantaged here as Eclipse is free so supporters have little economic incentive to develop partner programs that directly encourage Enterprise ISV’s to develop with Eclipse. Eclipse can
counter this by positioning itself as a “lock-in free” environment. This is an attractive value proposition to an Enterprise ISV who wants the widest possible market for its products and is loathe to become tied into a proprietary platform from IBM, BEA, Microsoft and others.

**Direct sales distribution.** Corporate IT Developers often purchase technology from a direct sales channel. These channels are difficult for companies to build and expensive to maintain. Since Eclipse is now embedded in the technology stacks of its major partners, every time a customer develops on top of the product from these companies, they will be using Eclipse by default. Thus, every sale IBM, SAP, or BEA makes is also an Eclipse sale. In this way, Eclipse benefits from the combined sales and marketing budgets of these companies which totals in the billions. Eclipse will therefore need to ensure that its IDE continues to be embedded in as many complementary IDE platforms as possible and that its partner program continues to expand.

**Retail & reseller distribution.** This is the primary channel to access the Mom and Pop and Department Developer segments respectively. Microsoft has highly developed indirect channels that Eclipse supporters likely cannot match. If a Red Hat-like Eclipse provider were to emerge and charge a small premium to package and distribute Eclipse as an IDE this deficiency could be addressed.

**Training & certification.** Corporate and Department IT segments place a premium on having access to training and certification for their developers. Developers value the certification for the salary premium they can command in the job market. Training and certification is currently far more evolved for Microsoft IDEs than it is for Eclipse. This may also be an opportunity for a Red Hat-esque company to develop a comparable certification program for more risk averse segments.

**Complementary development tools.** Corporate and Department IT segments are also consumers of non-IDE development tools like UML modelers and defect trackers that facilitate the development process. These non-IDE tools need to be very tightly linked with the core IDE. This market is currently a tossup between IBM’s extensive set of Rational tools based on Eclipse and Microsoft’s recent innovations in team development. IBM (and other Eclipse partners) will need to continue to invest in this area to gain revenue and competitive positioning vis-à-vis Microsoft.
Eclipse and the Free Rider Problem

As we see from the Eclipse value web, many industry players stand to benefit from the strategic shift Eclipse represents. We have also seen that to be successful across many major segments, Eclipse needs to develop an array of complementary assets. Since Eclipse is structured as a consortium and each partner has minimal capital at risk (albeit some more than others), developing expensive complementary assets for the Eclipse community is no easy task. This is essentially a free rider effect. Major players don’t mind donating a few developer resources, but lending millions of dollars in marketing dollars to the effort would benefit many while only costing one greatly.

For pure play IDE vendors such as Borland, the tendency to free-ride is even stronger. The commoditization of IDEs challenges their core “uniqueness” business model and is forcing them to down-size their efforts in developing their proprietary IDE and instead support Eclipse via a complementary asset business model. In this case, it is likely that Borland will significantly reduce their direct IDE investment efforts. While this increases the probability of advancements “around” the IDE, it potentially threatens advancements “within” the core IDE tool itself.

In the case of Sun Microsystems, who is one of the remaining Eclipse hold-outs, free-riding occurs in a unique context. While Sun’s participation in Eclipse would critically improve the consortium’s chances of success (previously discussed), Sun currently has more to benefit by remaining independent. By standing on the sidelines, Sun can free-ride Eclipse’s successes as growth of the Eclipse community creates greater demand for Java and in turn increases the value of Sun’s rights to the Java language and participation in the future.

There are two potential solutions to the free rider problem. Eclipse consortium members could revise the terms of membership to require meaningful financial, marketing, support and sales contributions to keep their position in the membership hierarchy. In the event this is too difficult to negotiate or contract, Eclipse will need to change the rules of the game and turn traditionally valuable complementary assets like partner programs and indirect distribution channels into liabilities for
companies like Microsoft. A difficult task but this would not be the first time Eclipse changed the rules of the IDE game.

Bibliography


