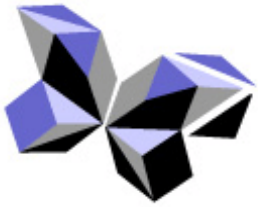

Digital Business Ecosystem

Contract n° 507953

WP 13: Business networks

D13.3: Final report on business networks

Information Society Technologies

Project funded by the European Community under the "Information Society Technology" Programme

Contract Number: 507953

Project Acronym: DBE

Title: Digital Business Ecosystem

Deliverable N°: D13.3

Due date: 31/7/2006

Delivery Date: 22/02/2007

Short Description:

The final report on business networks extends the case studies presented in the previous report, presents an updated qualitative model of platform evolution and identifies elements recurring in the established theoretical literature which support the empirical conclusions.

Partners owning: FZI

Partners contributed: FZI

Made available to: Public

VERSIONING		
VERSION	DATE	AUTHOR, ORGANISATION
1.0	**22/07/2007**	**TIM ROMBERG, FZI**

Quality check
Internal Reviewer: Nagaraj Konda (University of Central England)

Contents

1 Introduction

This third and final deliverable extends the previous work on business networks; extending the previous case studies on home video game consoles and Lotus Notes, the Apache web server is examined in the next chapter. It is not quite comparable to the other two in that Apache is not a commercial project; but, as will be shown, it draws on a community of commercial vendors who have been very instrumental in its success and are funding key members of the Apache foundation. On the other hand, it is interesting from a technical point of view because of its success in supporting the combination of independently developed modules. Based on all three case studies, a qualitative model of platform evolution is further developed in chapter 3. The fourth chapter reviews models of platform in the literature and identifies some theoretical conditions for the qualitative results of chapter 3.

2 Third case study: Apache web server

The Apache Web Server (also known as Apache HTTPD) is currently among the two most popular Web servers on the Internet (together with the Microsoft Internet Information Server)[1] and has been in this position for more than 10 years. It is released under the Apache License, an Open Source license which allows incorporating the software into proprietary products. The Web Server provides an API for plugging in various *modules*, and through this API, it provides a platform for a host of other Open Source projects (many of which are hosted by the Apache Software Foundation), as well as commercial products. The specific interest of this case lies in the fact that combining several independently developed modules on one server installation, so that they interact in a meaningful way, is not only theoretically possible, but actually practiced by typical Apache users. This is no small feat; we have described the difficulties this created on the Lotus Notes platform, and it is more difficult than only running several independent applications on the same hardware or software platform.

[1] Apache and IIS's relative position largely depend on whether one counts host names / IP addresses, or physical servers, and on other methodological choices. Most host names on the Internet today represent inactive or minimal Web sites, and one physical server can host hundreds of them. Cf. "Web Server Survey", "Hosting Provider Server Count" on http://www.netcraft.com, and http://www.port80software.com/surveys/top1000webservers/.

2.1 Timeline

In February 1995, the Apache Group was formed by a group of 8 administrators of the then-popular, free NCSA HTTPD server.[2] Through a mailing list, patches for its source code were exchanged for common fixes and enhancements.

In parallel, Apache Group member Robert S. Thau worked on an experimental rewrite coined "Shambhala" which emulated most of the NCSA server's functions, but was much more modular and resource-efficient. In June and July 1995, the group realized that this new implementation had much more potential than the original NCSA server.

Late november, 1995 saw release 1.0 of the new Apache server.

In March 1996, Apache became the most popular Web server according to Netcraft statistics (based on host names, not servers). This also marks the beginning of a slow but steady decline of all of the other Web servers, except IIS.

In November 1998, discussions started on Apache 2.0, the next major rewrite of the server, which would give even more flexibility for different types of modules and hardware platforms.

In June 1999, the Apache Software Foundation was incorporated as a non-profit corporation.

In March 2000, the first Apache conference (ApacheCon) took place, thanks to sponsoring by IBM and a few others.

In April 2002, Apache 2.0 was released.

2.2 The Apache ecosystem today

The Apache license, unlike other Open Source licenses, allows incorporating the Web Server into proprietary products and is therefore considered "business-friendly". Throughout its history, important contributions to the Web server and other Apache projects have been made by large and small technology providers. Important large providers are shown in table 1. Two smaller providers with fairly high presence are CollabNet (distributed software development solution) and Covalent (administration and monitoring tools for Apache servers).

Contrary to what one might expect, only one large vendor (Oracle) can be said to *rely* on the Apache Web server as part of a product (Oracle Application Server). IBM WebSphere, JBoss and BEA Weblogic are well-integrated and frequently *used together* with Apache by many customers, but these products do not depend on the Apache Web server to function; WebSphere can just as well be integrated with Microsoft IIS. The main interest for these vendors therefore does not lie in the technology of the Web

[2] National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, Hypertext Transfer Protocol Daemon. A "daemon" is a server process.

Table 1: Large tech vendors contributing to Apache

Vendor	Related product	Specific contributions
BEA	BEA Weblogic Application Server	Beehive project
IBM	IBM WebSphere Application Server	Sponsors many Apache developers, conferences, contributed XML tools
Oracle	Oracle Application Server	MyFaces project (Oracle ADF framework)
RedHat	JBoss Open Source Application Server	Tomcat project
Sun Microsystems	Java	Tomcat, Geronimo projects (free Java application server)

server itself, but in visibility and access to the community of developers and administrators around it. By donating code which is linked to a specific vendor-sponsored API (such as Oracle ADF), the API can achieve rapid popularity among developers. By creating a stream of projects and developers relying on the API, vendors can hope to channel part of them towards their high-end products. On the other hand, the vendor risks losing some existing customers to the free Open Source solution, which may be or become powerful enough for mainstream use. And since the API becomes open and well-documented, nothing will keep competitors for tapping into the same community, in the long run, by implementing it.

2.3 Architectural features

One of the great achievements of the Apache architecture, and one of the factors explaining its success, is that it has enabled modules to be developed very much independently from each other, in- and outside the Apache developer community, and today even typical installations contain several such modules which interoperate, often even in processing a single request, e.g.:

- mod_auth authenticates the user
- mod_perl executes a script program to produce dynamic content
- mod_deflate compresses the result

- mod_ssl encrypts it

etc.

We will use figure 1, which shows one view of the current architecture, to illustrate the most important concepts. The first rewrite ("Shambhala", or Apache 1.0), already introduced the main elements:

- Modules register themselves with the server.
- Request processing is broken down into several phases with semi-formally defined semantics. Each module is invoked for each phase (call handler), and can decide to handle it exclusively, cooperatively, or not at all.
- Modules register configuration commands and thereby receive their configuration from configuration files read by the server core (not shown in the diagram).
- Modules interact through centrally managed data structures with known semantics, such as for the request (request_rec), and for the server itself (server_rec).
- Modules can allocate resources in resource pools whose life cycle is managed by the server core.
- Modules can issue subrequests which run through a similar chain of phases as the main request (not shown).

The phases in Apache 1.x are ([Tha96], [Tha]):

- URI to filename translation;
- several phases involved with access control;
- determining the MIME type (e.g. HTML or JPG) of the requested entity;
- actually sending data back to the client;
- logging the request.

This is only a semi-formal description, but it is usually enough for a module developer to decide independently for which phases handlers must be implemented, and which part of the work gets done by which handler. The developer also knows the centrally managed data structures, their semantics, and has a rough idea about which parts of the structure are affected by which phase.

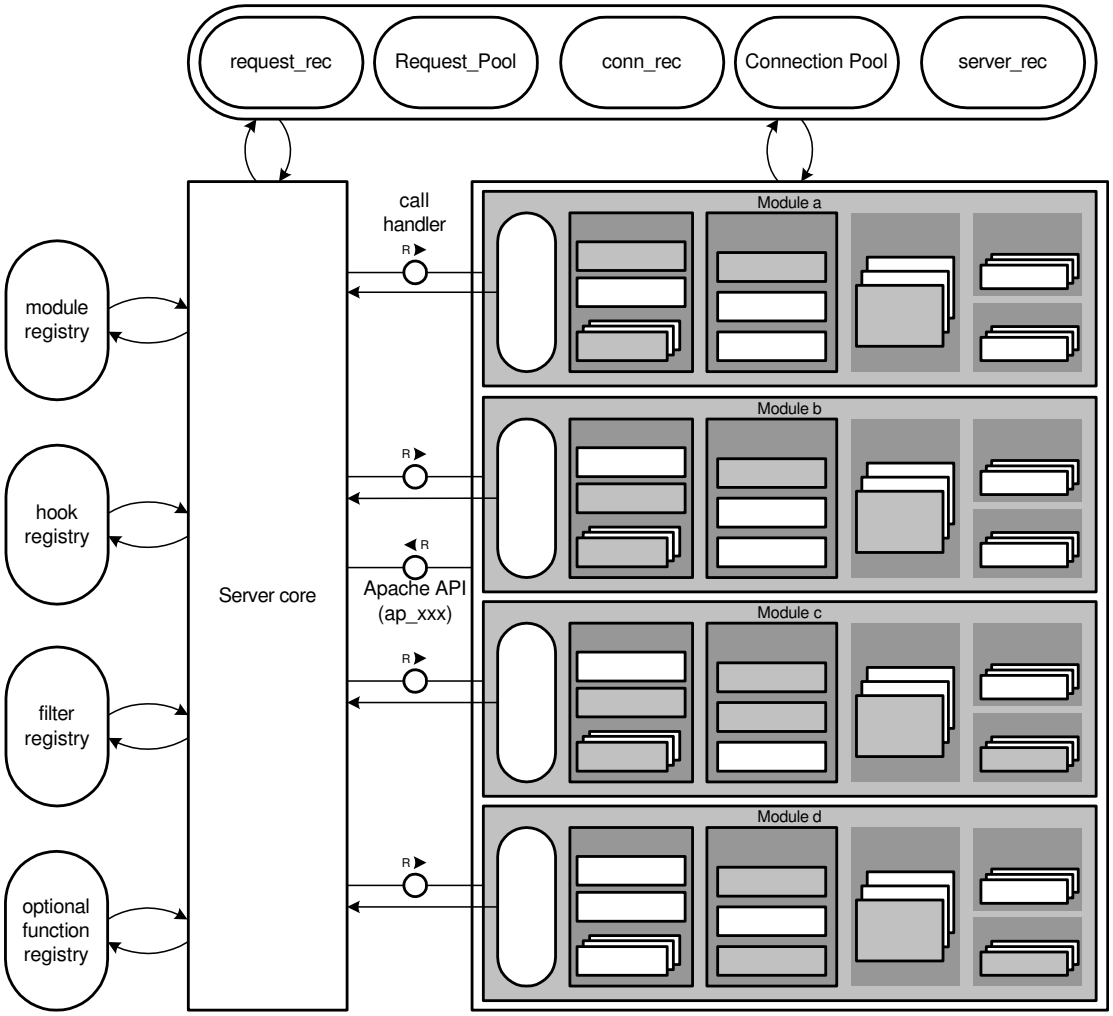

Figure 1: Interaction between modules and core in the Apache server. [GKKS04, chapter 3]

While the specification may be imprecise, an important force in resolving the semantic ambiguities is the set of standard modules included in the Apache release. The initial set of modules was designed to provide a precise drop-in replacement for the previous, monolithic NCSA implementation, and therefore quite extensive knowledge had already been accumulated within the Apache group about possible problems in the interaction or ordering of functionality.

Had the Apache server been designed from the beginning as a generic network server platform - which is what it could be seen as today - then the initial API would very likely have been much less concrete, perhaps only defining a generic "processing phase" concept without the semantics described above.

Nevertheless, as more and more modules were developed, and experience accumulated, the Apache 1.x design showed certain limitations, and the precise ordering of handler calls became a source of problems. The 2.0 design therefore introduced even more flexibility:

- Instead of a fixed structure which contains all the handlers, there is an extensible set of hooks that each module can hook into. The standard hooks still represent a centrally defined phase model of request processing. But a module can create new hooks and thereby meaningful events that other modules may want to react to.
- Requests and responses can be channeled through a chain of *filters*, like in a pipes & filter style.
- The variabilities of different operating system platforms with regards to multiprocessing are isolated into a wholly different type of module, the multiprocessing modules.
- HTTP becomes just one of many protocols supported by Apache.

It is beyond the purpose of this report to discuss these in more technical depth. But what is interesting is that in order to reach out to more applications and more hardware platforms, the scope of the server core itself has been *reduced*, while the server together with the standard collection of modules covers everything it did in the previous version, and more. Instead of one API, there are now at least three important interfaces: The successor to the module API, the multiprocessing module interface, and an operating system abstraction layer called Apache Portable Runtime, whose implementation has become a project of its own and is being used by a few other projects as well.

Roy Fielding, founding member of the Apache Group, summarizes the relationship between social organization and software architecture as follows [Fie05]:

> "What is common to the largest and most successful Open Source projects? A software architecture, designed to promote anarchic collaboration through extensions, while preserving control over the core interfaces."

3 Qualitative conclusion from case studies: An updated model of platform evolution

All three cases discussed in this work package have shown how successfully established platforms develop dynamics towards both the need for and the provision of inter-application interoperability standards.

To illustrate the quite complex dynamics of these inter-application networks above the platform, the following figure shows the platform and its relationships to the other players or technologies directly involved:

- The base technology on which the platform is implemented;
- the alternative, competing platforms;
- the complementers, or application developers; and
- users of the platform and its complements or applications.

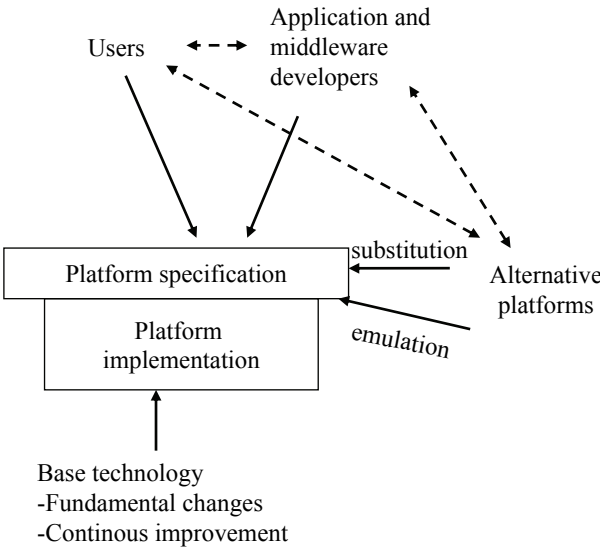

Figure 2: Four forces acting on a platform

This representation is inspired by Porter's five forces model ([Por80]), which would be a little bit difficult to apply to the software industry (who are "suppliers"? Where do the application developers fit?). It does not

show all the relevant factors for the continued success of a platform vendor, such as the very important issues of human resource management, research and development, and distribution.

The *base technology* is linked to assumptions made during specification and implementation of a platform. For software platforms, the base technology can consist of soft- and hardware. The progress of the base technology is a source of disruption, because it can invalidate these assumptions. In case of such a disruption, the entire technology stack upwards has to be recreated layer by layer. This happened to all but the last few generations of video game consoles, and it can still be observed in mobile computing. [GC02] address the issue of platform scalability in a case study of Intel.

Alternative, competing platforms are naturally a threat to the platform in question. Platform specifications compete for users and application developers ("substitution"); on the other hand, a well-established platform implementation can also be attacked by an alternative implementation which emulates its interface ("emulation") One of Atari's second generation competitors, Coleco, offered an adapter so its console could play games produced for the Atari 2600 (a strategy which [BF94] call "pesky little brother").

Application developers are of course vital for any real platform according to our definition. Platform vendors have to rally them early in the establishment of the platform ([BF94], [Hil97]). What is rarely mentioned is that companies acting as application developers often work on top of several platforms, sometimes by creating their own internal platform abstraction layer which reduces the cost of porting without eliminating it. Some can aspire to become platform vendors themselves.

In the case of video games, the need for modularization into engines, or middleware, and the game titles themselves became necessary because of the rising complexity of games (which is in turn a consequence of the success of video gaming) and a hardware technology which is very dynamic, but in the form of a continuous improvement. An engine could be made to scale better with this development than the console platform as a whole. But in contrast to the other cases, games only have to fit with the console (and the other environment), not with other games.

In the case of Lotus Notes, advances in the base technology were less relevant (although it cost Lotus its former flagship product 1-2-3). The need for an application core arose because of the success Lotus enjoyed with large customers. But neither the solution partners among themselves nor Lotus succeeded in establishing one.

The redesign of Apache certainly reflects the widespread success of it as a platform for module authors, and the fine-grained control needed to synchronize them. The new multiprocessing modules are partly a concession to an aspect of the base technology, in this case Microsoft Windows, whose multiprocessing model was not very well supported initially.

To summarize, there are many reasons why a given platform architecture, once established, develops into a more complex network based on specific common concerns and interoperability needs between applications which are not supported by the platform itself (although we cannot hope to prove that this always happens). This changes the nature of competition in the network. While the platform standards battle may be over, the platform vendor must pay closer attention to these developing networks.

If several platforms have survived, firmly entrenched in their respective community, then at this point it is probably unfeasible and too costly to continue the platform battle with full energy. Instead, if barriers between platforms are maintained, there is the danger of creating a great business case for cross-platform middleware in various application areas, which would undermine the negotiating power and sales relationship of the platform vendor to his customers. It could therefore make more sense to establish interoperability with other platforms to some degree. In recent Computing history, Java was initially sold by promising to bridge the Unix and Windows worlds (plus mobile devices). Now that Java is recognized as a platform in itself, XML-based Web services promise to bridge the Java and Microsoft .NET world. This, along with the ability of application developers to support several platforms, may explain why, in spite of tipping, in many areas we find not one, but two or three well-established platforms. (See summary in table 2).

The platform vendor or sponsor will benefit from getting involved in the user community structuring and application-middleware-building process himself in order to maintain significance. Large or otherwise very credible vendors can considerably accelerate the creation of application interoperability standards, as Lotus tried belatedly and unsuccessfully, and Apache did quite successfully. As a result, the platform may have to split into several levels - as shown in the Video Game and the Apache case.

4 Game-theoretic analyses

This chapter will try to relate the empirical findings to game-theoretic models of software markets and platform competition. The first subsection summarizes how these models were developed since the 1980s. The next subsection discusses two models of platform competition in detail. A more generic framework of software platform markets is developed in the third subsection. A simple model is presented to support the empirical findings of the last chapter, concerning the platform vendors. Finally, the cooperation between application vendors is discussed, linked to the models developed in the earlier deliverables.

Table 2: Summary of cases

	Home Video Games	**Lotus Notes**	**Apache HTTPD**
Why the need to interoperate?	Rising complexity of games, Dynamic but predictable base tech.	Core domain objects	Modules perform complementary functions
New interop. specs initiated by	Mostly game software developers	First solution partners, then Lotus, then solution partners...	Apache project (for HTTPD core)
Challenge of base technology	New hardware acceleration schemes invalidate engines	(none identified)	New hardware and operating system concepts
Number of major competing platforms	3 (excluding PC and mobile)	2	2
(Possible) outcome	Stronger position of game software vendors	Notes often used in basic configuration only, not mission-critical	growing support by industry; but also: becoming a mere engine for running higher-level runtimes

4.1 Review of theories of compatibility, components, and platforms

Compatibility and compatibility standards are a prominent topic in the overall network economics literature. One research strand is concerned with the direct network effects of certain network goods, where an increase in adoption directly benefits each buyer. A well known example is Metcalfe's law for telecommunication networks, which states that its value for each participant increases linearly with the number of other participants, and hence the total value of the network of size n is proportional to $n(n-1)$. When customers face the choice of joining one of several possible networks, the decisions of the early buyers can greatly influence the overall outcome ([Art89]). The market tends to tip towards one technology, and not necessarily the best one. By choosing a network, customers bestow an external utility upon the other customers in the network and therefore, under some circumstances, not as many customers join as would be socially optimal.

Vendors in these "one-way networks" (as [EW95] call them) face the choice of making their products compatible with their competitors. [BF94] examine situations with two vendors in which this is desired by both vendors, undesired by both vendors, or desired by only one vendor. [KS85] compare the possible equilibria of n vendors choosing their output under different compatibility regimes. A rather intuitive result is that when a sub-coalition of vendors decides to make their products compatible, they can increase their total market share while the market share of all other vendors combined shrinks. However, their model does not account for heterogeneous customer preferences regarding different vendors or compatibility standards. In practice, while standardization allows for large networks and high network utility, it also often reduces technical variety and can slow innovation, at least in the standardized component (cf. [FS86]). A compromise could consist of differentiated technologies which maintain a certain degree of compatibility, e.g. through adapters (converters). [FS92] examine the effects of such converters and the incentives to provide them. The converters are "imperfect" in the sense that the network utility derived from a member of the other network is still lower than that from a member of one's own network. The somewhat counter-intuitive result is that these converters often lead to lower overall social welfare than a situation without converters. This is because they can lead both buyers and vendors towards two separate networks where without converters one large network would have formed.

[MR88] started another strand of analysis dealing with compatibility between complements. They analyze strategic interaction between two system vendors who each offer a system of two components (e.g. CD player and amplifier). In the first stage of the game, vendors decide whether they

allow buyers to *mix and match* components. In the second stage, they set prices. Buyers have heterogeneous preferences regarding the components; there are as many who would prefer a hybrid system as those who prefer buying from only one vendor. The model predicts that prices are higher under compatibility - a phenomenon similar to the double marginalization of vertical monopolists ([Cou38]). Therefore, those buyers who do not want to mix and match are worse off under compatibility. Equilibria of the game and welfare considerations depend on the exact parameters, but in general, if there are no extra costs for the development of an interoperability standard, compatibility is socially beneficial, good for vendors, and an achievable equilibrium.

More generalized mix-and-match models have been proposed by [Eco89], [ES92] and [MR92]. [MR88] have interpreted their analysis as one of "compatibility without network externalities", since buyers do not benefit from more other buyers buying the same product or system. In the earlier analyses, buyers benefited from (full) compatibility because it meant a larger network; here some buyers benefit from being able to choose a system which is closer to their preferences. The *externalities* discussed in the mix-and-match models concern the effects that price changes in one good (component) has on the demand of another good. They are not demand-side economies of scale. [EW95] offers the interpretation that the earlier models took network externalities as given (a "Macro" approach) while the latter try to model them as the result of a technology structure ("Micro" approach). This is plausible when one thinks of PCs - a network good from a Macro perspective because it is clearly beneficial to buy a PC with a large user base. But from a Micro perspective, there are many individual benefits arising from other users having the same model: More available technical support from professionals or friends/colleagues, exchange of files with friends/colleages (including free or pirated software), and more software variety. Software is clearly a complement good to the PC hardware; both together form what Economides calls a "two-way network", and the network effects are "indirect" rather than direct as in the case of telephones.

In the early mix-and-match models, the number of varieties of each *component* was fixed (2); only the number of possible *systems* could be influenced. The effect just mentioned of hardware standardization on software variety was therefore not included. [Eco91] and [CG92] attempt to address this effect. As they come closest to addressing an economic analysis of software platforms, the next section will treat them in depth.

4.2 Platforms, platform compatibility, and free entry

In both models there are two types of goods. There are two varieties of the first type (A_1 and A_2), and the number of varieties of the second type

B is to be determined, e.g. through free entry. Each product is offered by one vendor and vice versa. In [CG92], demand is based on a utility function of individual buyers, who choose exactly one of the two varieties of the first type (explicitly called "hardware"), according to a Hotelling-like preference model. Their utility from the second type of product (software) follows a CES (constant elasticity of substitution) function. A buyer whose preference for the first type is a distance t away from the chosen product derives utility

$$U(x_1, x_2, ..., x_N, t) = (\sum_i x_i^{1/\beta})^\beta + \phi - kt, 1 < \beta < 2 \quad (1)$$

when buying quantities $x_1, ..., x_N$ of the second type products, where ϕ, β, and k are constants. This is equivalent to the assumption that buyers buy a little bit of every product of the second type — which seems unrealistic when one thinks of the second type of product as "software" or "applications", but makes the formal treatment with a variable number of products easier. All buyers will in fact spend the same proportion of their software budget (total budget minus hardware price) on a given software product. The parameters are restricted such that all buyers will buy one of the two platforms, along with some software.

[CG92] is mainly concerned with entry and the viability of two incompatible platforms vs. tipping towards one platform — compatibility between the platforms is not an option. The timing has three steps: First, software vendors decide whether to enter the market at all, then decide on one hardware platform. (Actually, this should lead to the same result as software vendors entering and choosing a platform in one step, as long as they already have a fairly good idea of how many vendors there will be in the end.) As the third step, buyers choose hardware and their level of software demand. The platform prices are exogenous and equal in this model. Software prices are assumed to be determined by monopolistic competition — this is the equilibrium when vendors do not take into account an indirect effect of their pricing on demand, and leads to a price of βc_B where c_B is the variable cost, regardless of the number of vendors. In a related paper ([CG93]), the authors discuss the Bertrand equilibrium, where vendors do take into account the indirect effect. The price then decreases with an increasing number of vendors, as we should expect, and asymptotically approaches βc_B as their number gets large. Nevertheless, since the variable cost of software production is typically negligible, the result in the original model seems to contradict observation.

The assumption of a constant price for software simplifies the discussion of the software vendors' choice of a platform, and of buyers choice of a platform. If the price depends on the number of software vendors on the same platform, each software vendor's profit would still be roughly equal across platforms in equilibrium, and, under free entry, close to zero

(the remaining difference would only be an integer effect, as in the original model). However, buyers would have to take into account the different software price levels when choosing a platform.

The number of free entry software firms is easily determined in the Church and Gandal's model because they assume equal hardware prices. There are then three candidate equilibria: Standardization on platform A_1, standardization on platform A_2, or equal division of software vendors and buyers between the two platforms. Depending on parameters, a subset of these equilibria actually exist. For example, strong hardware differentiation, high software costs, and a fairly low preference for software variety are all factors that tend to lead to an equal division between the two platforms. But in all three candidate cases, a given number of software firms will lead to the same profit per software firm. If the two platform prices were different, this would not necessarily be the case: For a given number of software firms, some equilibria could lead to higher software profits than others, meaning that depending on the equilibrium, a higher or lower number of software firms could be sustained.

Despite these limiting assumptions, the model has made a great achievement in formalizing the relationship between the adoption of a platform by software vendors and by buyers, and the tradeoff, for software vendors, between the indirect network effect - more buyers - and the competitive effect of more other software vendors on the same network. For buyers, the tradeoff is one between the variety of software on a platform and individual preferences for a platform (but not, as mentioned, prices of hardware or software).

[Eco91] ultimately addresses the question of whether platform vendors should make their platforms compatible. This model is a more direct descendent of [MR88]: The structural difference lies in the separation between vendors of the two component types (in [MR88] they are vertically integrated), and in the variable number of type B products. In the case of incompatibility between the "platforms" A_1 and A_2, each vendor of type B products produces two varieties, one for each platform. The fixed cost for the two varieties is higher than the fixed cost in case of compatible platforms. There are three stages of the game: In stage 1, the type A vendors decide if they want to be compatible with the other vendor. Compatibility requires both vendors to agree to it. In stage 2, type B vendors enter. In stage 3, all vendors choose prices.

Total demand for a system of two components is a linear function in prices. For systems built with A_1 and B_j, demand is:

$$x_{1j} = a - b(2n-1)(p_1 + q_j) + c\sum_{k\neq j}(p_1 + q_k) + d\sum_{k}(p_2 + q_k) \qquad (2)$$

when platforms are incompatible, and

$$x_{1j} = a - b(2n-1)(p_1 + q_j) + c\sum_{k\neq j}(p_1 + q_k) + c(p_2 + q_j) + d\sum_{k\neq j}(p_2 + q_k) \tag{3}$$

when they are compatible, where p_i are the prices of type A, q_j the prices for type B products, $a, b, c, d > 0$ are constants, and n is the number of type B products. The logic behind the different coefficient on $(p_2 + q_j)$ is that under incompatibility, this stands for a system which differs in two components — because it includes a distinct variety of B_j for that platform. Parameters are restricted by $b(2n-1) > c(n-1) + nd$, so that an equal increase in the price of all systems leads to decreasing demand in each system. The author probably forgets mentioning that $b(2n-1) > cn + d(n-1)$ is also required because of the compatible case; we would also expect $d < c$, although this is not made explicit anywhere.

The reasoning is as follows: Bertrand price competition among type A and type B vendors leads to equal prices, quantities, and type A profits across compatibility regimes. Given a certain number of type B vendors, their profits are lower under incompatibility because of the higher fixed cost. Free entry will therefore lead to fewer type B vendors under incompatibility than under compatibility. This result perfectly agrees with observation on typical platform markets, as mentioned at the end of the last section (p. 13), although the argument is usually more complex: When several incompatible platforms compete, both application developers and users tend to wait because choosing the wrong platform can be very costly.

For the compatibility decision, all depends how the type A vendors' profits evolve with the number of type B vendors (variety), n. [Eco91] argues that when total demand for type B products does not significantly increase with n, then profits of type A vendors decrease with n. Therefore type A vendors prefer incompatibility. On the other hand, if total demand scales up with variety, profits of type A vendors may also increase, and then they prefer compatibility.

The model has made an important contribution in being probably the first to address the application variety effect of platform standardization using a free entry model. By assuming application vendors to always develop for both platforms if they are incompatible, it constitutes the matching counterpart to [CG92]'s assumption which requires all application vendors to choose exactly one platform.

However, the model is less satisfactory in some respects than [CG92]. There are some general issues, and some regarding its application to typical platforms. After all, it is a working paper, and the fact that it was not developed in subsequent publications indicates that the author became aware of the following difficulties himself.

First, notice that the demand function 2 with given coefficients only

applies to a specific n. For variable n, decreasing n by one should be equivalent to raising the price of the nth product towards infinity. With a linear demand function, this is not possible.

Therefore additional assumptions have to be made as to what happens with increasing n. For example, to make the free entry logic work, it has to be assumed that type B vendors' profits decrease with n. It becomes hard to verify what the interaction between the various assumptions is and how they can be simultaneously fulfilled. The discussion of entry and the effects of variety, based on scaling factors, is detached from the more precise mathematical derivation of the Bertrand equilibria.

Second, while the use of coefficients makes the demand function quite general, it is quite an assumption that they are not influenced by the choice of a compatibility regime. The demand function is based on a certain symmetry between the two components which seems inappropriate when applied to platform/application markets: The price for the "different application, same platform" system having the same coefficient as the "same application, different platform" system. (A customer will usually only buy one platform, but several applications.)

Finally, as in many other models, individual buyers' demand and buyers' utility are not expressed in this demand function, therefore it is not sufficient for a discussion of buyers' "adoption strategies", nor of social welfare.

4.3 Framework for platform models

It should have become clear that the two models discussed in the previous section are built on a number of important assumptions — some of them chosen in order to facilitate a formal treatment, all of them an opportunity to consider alternatives. The models often consider the extremes, because they are simpler and more instructive to discuss. There will therefore never be a convergence on "reasonable" assumptions, but there are recurring themes and issues. The framework illustrated in figure 3 includes the most common dimensions which determine models of economic interactions between platform vendors and application vendors. This framework allows us to qualitatively discuss effects of, for example, platform compatibility.

Exact game-theoretic models which take all the aspects into account would be far too complex. When analysing a real-world case, it is therefore often useful to first discuss it qualitatively in all its diversity, then try to construct a simpler quantitative model which concentrates on selected effects.

There are three roles: Platform vendor, application vendor, and customer. Therein lies a first simplification, of course, since the case when different players act as platform vendors and application vendors appears

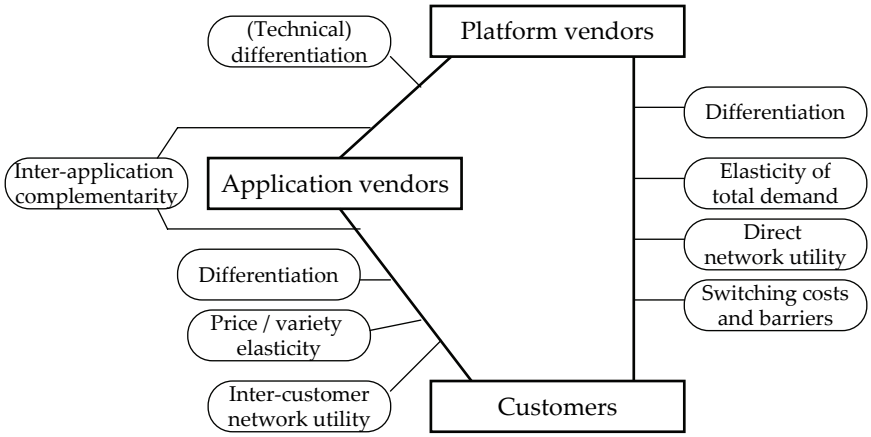

Figure 3: Common dimensions in platform models

more typical than a case when one player plays multiple roles. The dimensions concern the relationships between these roles (not between individual representatives).

Platforms and customers Platforms are in some way or another chosen by customers — in reality they are often sold in association with other products, such as certain applications or with hardware; but customers are normally still aware of the platform they are buying. It is typical that customers can or want to only run one platform, they commit themselves to it. The static platform models discussed above do not consider switching at all. When platform implementations are incompatible, switching entails a significant switching cost per installed application, but even if they are compatible (i.e. existing applications can be run on other implementations), there may be other switching costs related to administration, backups, data conversion etc.

As with other technologies, there may be a *direct network utility* associated with choosing a popular platform; for example, more available support from friends and professionals. The models in the mix-and-match and platform tradition were not concerned with these and highlighted the indirect network effects related to the complementary goods instead. [CG92] showed how a tipping effect towards one platform can be explained by the indirect network effect, of customers following application developers and application developers following customers, alone.

The elasticity of total demand with respect to (hedonic) prices will play a role in the discussion of platform compatibility below. [CG92] and [FS92] are examples of models where total demand is completely inelastic, or only the "inelastic region" of the parameter space is considered: all customers always choose one of the two available (platform) products. [MR88] con-

sider both elastic and inelastic regions: Below a certain hedonic price, all customers buy one system. In [KS85], there is an infinite pool of potential customers. On the other hand, in all models the (platform) buying decision is 0-1, where in reality customers might choose between different versions, pay per-processor licenses, etc.

Because of the "natural monopoly" tendencies brought about by direct and indirect network effects, the question is how multiple platforms are possible in the first place. When discussing platform competition, we normally want to weigh these effects against other effects. It would not be very interesting nor realistic to discuss models where the first mover always wins. The answer lies in differentiation. Platforms can differentiate themselves in four fundamental ways, namely with respect to:

1. applications,
2. a quality dimension (vertically),
3. an independent dimension relevant to customers (horizontally), and
4. customer networks.

With respect to different *application* classes, platforms may differ in their technical advantages for the implementing them. For example, the current Nintendo Wii console with its 3D motion-sensitive controller, but relatively limited processing power is good for rather simple sports games, while the current Microsoft Xbox 360 lends itself to more complex games with dozens of animated characters, possibly played by as many other human players connected through the online gaming service. *Vertical* differentiation is differentiation with respect to a *quality* dimension which is valued by all customers, but with different priority. The analysis of oligopolistic competition with quality was introduced by [GT79]. Quality is typically linked to costs (e.g. in R&D), and only serves as a differentiator if a single vendor cannot efficiently offer all desired qualities at the same time. [MR88], [CG92] and [FS92] have used *horizontal* differentiation following [Hot29], where customers' preferences are localized on a continuous spectrum of some sort, and vendors either choose or are exogenously located at certain positions in that spectrum. Finally, vendors can try to penetrate *subnetworks of customers* who are closely related. The Apple Macintosh, for example, successfully established a strong position in the Desktop Publishing area in the late 1980s, based on its special capabilities then, and has maintained a relatively high penetration in advertisement and multimedia agencies to this day, despite the fact that PCs now have roughly equivalent capabilities.

From a formal point of view, differentiation also provides a way to make demand a continuous function of price; price (Bertrand) competition being

more appropriate than output (Cournot) competition in the case of software. Of the models mentioned in this chapter, only [KS85] uses output competition, and consequently does not model any differentiation.

Platforms and applications Application vendors are depicted vertically between platform vendors and customers, for they, just like customers, must choose one or several platforms to build their products on, and frequently act as channels for the platform itself. Technically, a platform fulfills several distinct functions with respect to applications:

1. the platform as an API to access the lower-level functionality provided by platform implementations; this coincides closely with
2. the platform as a run-time environment chosen and installed by potential customers of the application; to an application vendor, these different platforms define different markets and often channels;
3. the platform as a programming environment, providing the low-level building blocks (e.g. programming language) from which the applications are built;
4. the platform as an inter-application communication mechanism.

In today's mainstream computing, speaking of "the platform" is often an oversimplification because each of these functions can now be provided by different technologies: The *OpenLaszlo* framework used in the DBE project, for example, provides a programming model based on the language *JavaScript*. Applications are then compiled and delivered to the client in several possible editions, for example one which uses the *Macromedia Flash* API and run-time environment. For inter-application communication, yet other technologies, such as *SOAP* can be used. Economically, API and programming environment constitute hard-to-change investments (high switching costs), especially the latter. The run-time environment is related to the network of customers, and the inter-application communication mechanism to the network of complements (if these exist). Note that using the same communication mechanism is a necessary, not a sufficient condition for two applications to interoperate. Still, all four aspects are strongly dependent on each other. In earlier hardware generations, such as the *IBM AS/400* (today's *iSeries*), all aspects were managed as one platform by one platform vendor.

The aspect of technical differentiation of platforms with respect to applications was already mentioned in the previous section. Vice versa, application vendors can choose to differentiate with respect to platforms, since platforms (as run-time environments) define customer networks, and an application developer incurs fixed costs for becoming productive in the APIs and building blocks of a platform,

Applications and customers Economically, the need for interoperability between applications appears as a characteristic of the utility function, where the consumption of two interoperable applications together leads to higher utility than the sum of its parts — they are complements. In the extreme case, utility can only be derived from systems of several applications, as in [MR88]. This utility often appears as an argument in platform vendors' advertisement: "Using our platform, you can mix and match various applications". Recently, for example, *SAP Netweaver* and *IBM Websphere* are sold as solutions to integrate all applications in an enterprise on one *Enterprise Service Bus*.

The market for applications is almost by definition a highly differentiated market — if there were only one application, there would be no reason to separate it from the platform in the first place. Horizontal and vertical differentiation (mentioned above) can be observed with applications, too. [CG92] assumes monopolistic competition, which closely describes the situation for very specialized application vendors. A big difference with platforms lies in the fact that customers often want more than one application. They may even have a taste for variety: Video game customers who have just bought a car racing game (because that is their favourite category) may then be more inclined to buy a game which is quite different from car racing, rather than similar. The taste for variety in the sense of the elasticity of total demand with respect to the number of available applications was an important parameter in both [CG92] and [Eco91].

Finally, an application can generate direct inter-customer network utility. In the case of video games, online gaming has been changing the industry, since there are now a few titles or series which bind a significant player community. This kind of network utility tends to limit variety by boosting a few popular titles.

Before using this framework to discuss platform evolution and case studies, a much simpler model will be presented to provide a basic argument for one of the main conclusions from the case studies.

4.4 Evolution of platform competition

One of the main conclusions from the case studies was that the nature of platform competition changes as a platform market matures. As with any technology, there is an era of ferment, where very different concepts are tested on the market. When a clearer understanding has emerged for a platform category, a battle for dominance between several incompatible platform standards often arises. When there are no significant existing networks of complements and customers, this competition will concentrate on features and price. As the networks grow, so does their importance for subsequent growth. Due to tipping, few offerings survive. If there are several survivors who succeed in building significant application and

customer networks, as these networks mature, application developers and customers complain more and more about the interoperability gap between the networks, and may start devising ways to bridge it. Since each surviving platform vendor is now in a stable position, it may be wiser at this point to lower the interoperability barriers, in order to avoid the creation of a new higher-level platform by some third party.

This evolution can be illustrated by a simple sequential game: Very much like [FS92], two platform vendors compete for members of their network (the argument can be made both for application developers and for end customers) whose preferences are distributed evenly between them, on a line of unit length, with those preferring platform A at $x = 0$, those preferring platform B at $x = 1$. Although in addition, their maximum stand-alone utilities y are distributed, so that total demand is elastic. Each member benefits linearly from the size of the chosen network. Therefore, a potential member located at (x, y) enjoys a utility (profit) of:

$$u_{\mathrm{A}}(t) = y - kx + gn_{\mathrm{A}}(t) - rp_{\mathrm{A}}(t_0), \text{ when choosing platform A in } t_0 \quad (4)$$
$$u_{\mathrm{B}}(t) = y - k(1-x) + gn_{\mathrm{B}}(t) - rp_{\mathrm{B}}(t_0), \text{ when choosing platform B} \quad (5)$$

for all periods $t \geq t_0$, and zero utility for periods before the investment period t_0, where $k > 0$ is the strength of the horizontal differentiation, $0 < g < 1$ the strength of the network effect, $n_{\mathrm{A}}(t)$ the size of network A at the beginning of period t (including all members who have joined in periods 0 up to $t - 1$), and $0 < r < 1$ the interest rate, so that spending one unit of money at some time t is r more expensive (in terms of that period's utility) than spending it at $t + 1$.

By spreading out the investment cost over the utility of subsequent periods, it becomes rational, under certain assumptions, to invest in a platform once its u is greater than zero; and if this is true for both, choose the one with the greater current u. The assumptions are:

1. Once a platform is chosen, switching is not possible. This also ensures that the chosen network will not be shrinking, and the subsequent periodic utilities will be non-decreasing.

2. Potential members do not make bets on a platform, which looks less attractive now, becoming more attractive tomorrow.

3. They also do not bet on price changes.

When taking "members" as "application developers", the inability to switch can be explained by their software artifacts being made of the building blocks of the platform. Implementing on top of another platform would mean to start from scratch, and there is no reason to stop selling the application on the original platform.

The potential members are distributed evenly with density 1 on an infinite rectangular area limited by $x = 0$ and $x = 1$ on the left and right, and $y = Y_0$ on top. Negative y simply means that a potential member cannot achieve positive utility without a certain network size.

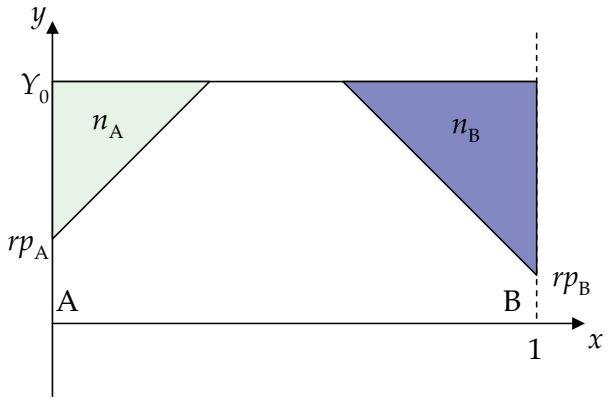

Figure 4: The sequential game after the first period

Figure 4 shows the situation after the first period. All potential members located above the line $y = rp_A(0) + kx$ are joining platform A. Therefore

$$n_A(1) = \frac{1}{2k}(Y_0 - rp_A)^2 \tag{6}$$

as long as the two regions do not meet each other: $r(p_A + p_B) > 2Y_0 - k$. In the next period, even without any price changes, new members will join the two networks because they are now more attractive.

It is easy to see that this game has the same tendency of tipping towards one of the two platforms as the sequential games used for explaining tipping in the literature (cf. [Art89]); only here it is not a sequential arrival of individual customers, but a pool of potential members with distributed characteristics.

This can be illustrated using figure 5. Platform A has succeeded in attracting members to the left of $x = x_0$, while platform B has attracted members to the right of $x = x_0$, as shown. Their next most closely disputed member sits at $(x_0, \tilde{y})$. In order to keep dividing members along the line $x = x_0$, their prices must fulfill the condition:

$$r(p_A - p_B) = (2x_0 - 1)(g(Y_0 - \tilde{y} + \frac{k}{2}) - k) \tag{7}$$

which means, in the depicted case of $x_0 > \frac{1}{2}$, that, as $\tilde{y}$ moves downward, B must charge lower and lower prices in relation to A. It becomes

easier and easier for A to price B out of the market.

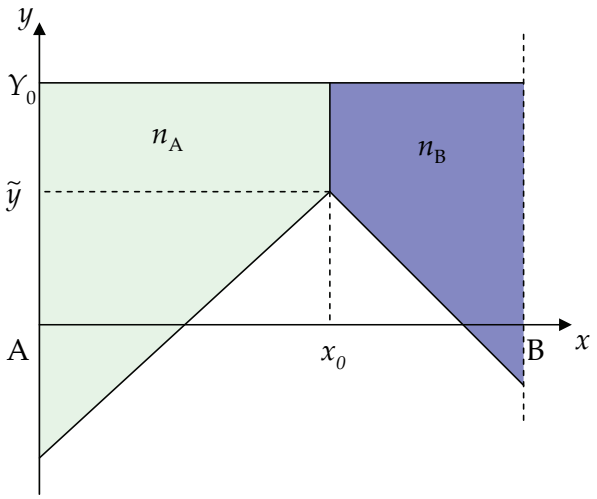

Figure 5: Division along the line $x = x_0$

We can therefore expect the competition to be rather fierce in the beginning, with the usual penetration pricing, hoping to be able to charge higher prices once a dominant position is reached. In the long run, the most likely outcome (how likely will depend on the parameters) is one dominating platform. A less likely, but possible outcome, is that both platforms survive, with market shares close to 50% each. In any case, there is a natural limit to the growth of the networks. In the case of a 50/50 share (and equal prices), we have

$$n_A(t+1) = \frac{1}{2}(Y_0 + g n_A(t) - r p_A) - \frac{k}{8} \tag{8}$$

and the network sizes will gradually approach (solving for $n_A(t+1) = n_A(t)$)

$$n^{\max}_{50/50} = \frac{Y_0 - rp - \frac{k}{4}}{2-g}. \tag{9}$$

For a monopolist, the limit would be

$$n^{\max}_{\text{monopoly}} = \frac{Y_0 - rp - \frac{k}{2}}{1-g}. \tag{10}$$

Once close to that limit, only lower prices will attract new business. At this mature stage, and only if in fact both platforms have survived, a two-way converter can enable both to expand their business beyond that limit. Following [FS92], a two-way converter will enable members of one

network to benefit from a certain fraction $q, 0 < q \leq 1$ of the network utility of the other network. The new utility of a member of network A is therefore

$$u_{\mathrm{A}}(t) = y - kx + g(n_{\mathrm{A}}(t) + qn_{\mathrm{B}}(t)) - rp_{\mathrm{A}}(t_0). \tag{11}$$

In the case of both platforms having 50% market share, the new limit is

$$n^{\max}_{50/50,\,\text{converter}} = \frac{Y_0 - rp - \frac{k}{4}}{2 - g(1+q)}. \tag{12}$$

A converter will tend to diminish any advantages one platform has in terms of market share and competition will once again focus on differentiating features. Therefore we can expect this scenario to happen only if market shares are very balanced, and there is sufficient differentiation possible after a converter is introduced.

In relation to the framework in figure 3, we can say that several conditions are necessary so that the introduction of platform compatibility is attractive to platform vendors at some mature stage, as predicted by our hypothesis:

- total demand for platforms must be elastic;
- there is a direct (platform-related) or indirect (application-related) network effect. In the latter case, for example, the compatibility allows new mix-and-match combinations of applications;
- differentiation between platform vendors is possible even after the introduction of compatibility;
- platform switching costs are high enough so that vendors do not have to fear complete erosion of their established developer and customer base in a price war.

4.5 Joint development of components

The previous reports presented a model of coalitional licensing of components between software vendors, based on a combination of cooperative and non-cooperative game theory. A general observation based on a number of examples was that equilibria in which licensing takes place will are only possible when vendors have asymmetric costs. It is also quite straightforward to explain analytically why such licensing does not take place with symmetric...

For the analytical treatment, we begin by considering only two vendors (1 and 2) and two products (a and b). Both vendors have the option of entering none, either one, or both of the product markets. Again, abstracting away from any particular assumptions about the demand, revenue from

product i when there is only one vendor of this product is $r_i(1)$, and $r_i(2)$ for each vendor if there are two vendors. We should expect

$$2r_i(2) < r_i(1) \tag{13}$$

which means that the second vendor cannot increase total demand to an extent where it will offset the revenues lost by price competition. Vendors are assumed to either have equal cost, or each have an advantage in making one product. In this case the vendor with the advantage in making a will be labelled 1 without further loss of generality. c^i_{a} is vendor i's cost of making a; c^i_{b} his cost of making b, and c^i_{ab} the cost of making both, with

$$e^i = c^i_{\mathsf{a}} + c^i_{\mathsf{b}} - c^i_{\mathsf{ab}} \geq 0 \tag{14}$$

the non-negative economy of scope when making both products. Of course, it is never cheaper to make both products than only one:

$$e^i < c^i_m, m \in \{a, b\}, i \in \{1, 2\}. \tag{15}$$

It is attractive for both vendors to be a monopolist in either product: $r_m(1) > c^i_m$. The stronger product of a vendor always delivers a higher or equal margin than the weaker product:

$$r_{\mathsf{a}}(k) - c^1_{\mathsf{a}} \geq r_{\mathsf{b}}(k) - c^1_{\mathsf{b}} \quad \text{and} \tag{16}$$

$$r_{\mathsf{b}}(k) - c^2_{\mathsf{b}} \geq r_{\mathsf{a}}(k) - c^2_{\mathsf{b}} \quad \text{for} \quad k \in \{1, 2\}. \tag{17}$$

A vendor will suffer from a negative payoff when only selling the weaker product under competition:

$$r_{\mathsf{b}}(2) < c^1_{\mathsf{b}}. \tag{18}$$

In the first stage, vendors decide cooperatively whether to make a shared component z which can be used for both products. This may or may not be the same technical component as the one responsible for the economies of scope above. When z is available, the remaining cost for making a is denoted as $c^i_{\mathsf{a}|\mathsf{z}} = c^i_{\mathsf{a}} - \Delta_{\mathsf{z}} c^i_{\mathsf{a}}$. We assume that building the component is worthwhile only if it is actually used by both vendors to build both products:

$$0 < \Delta_{\mathsf{z}} c^i_m \leq c_{\mathsf{z}} < \Delta_{\mathsf{z}} c^i_m + \Delta_{\mathsf{z}} c^j_n, n \neq m, i \neq j \tag{19}$$

$$0 < \Delta_{\mathsf{z}} c^i_{\mathsf{ab}} \leq c_{\mathsf{z}} \tag{20}$$

The total economies of scope (including the contribution of the component) do not increase when a shared component is built:

$$c^i_{\mathrm{ab}|\mathrm{z}} - c^i_{m|\mathrm{z}} \geq c^i_{\mathrm{ab}} - c^i_m, m \in \{\mathrm{a}, \mathrm{b}\} \tag{21}$$

$$\Leftrightarrow e^i - e^i_{|z} = \Delta_z e^i \geq \Delta_z c^i_m \tag{22}$$

In the second stage, vendors decide non-cooperatively which products they build and offer. Each vendor has four strategies. The payoffs are given by table 3. In case a shared component is developed, the $c^i_{m|\mathrm{z}}$ apply, and the Π_i denote the payoff from the second stage. The payoffs of the overall game have to take into account the cost of the component shared in some way, yet to be determined, between the vendors.

Vendor 1	Vendor 2	Π_1	Π_2
$\emptyset$	$\emptyset$	0	0
	a	0	$r_{\mathrm{a}}(1) - c^2_{\mathrm{a}}$
	b	0	$r_{\mathrm{b}}(1) - c^2_{\mathrm{b}}$
	ab	0	$r_{\mathrm{a}}(1) + r_{\mathrm{b}}(1) - c^2_{\mathrm{ab}}$
a	$\emptyset$	$r_{\mathrm{a}}(1) - c^1_{\mathrm{a}}$	0
	a	$r_{\mathrm{a}}(2) - c^1_{\mathrm{a}}$	$r_{\mathrm{a}}(2) - c^2_{\mathrm{a}}$
	b	$r_{\mathrm{a}}(1) - c^1_{\mathrm{a}}$	$r_{\mathrm{b}}(1) - c^2_{\mathrm{b}}$
	ab	$r_{\mathrm{a}}(2) - c^1_{\mathrm{a}}$	$r_{\mathrm{a}}(2) + r_{\mathrm{b}}(1) - c^2_{\mathrm{ab}}$
b	$\emptyset$	$r_{\mathrm{b}}(1) - c^1_{\mathrm{b}}$	0
	a	$r_{\mathrm{b}}(1) - c^1_{\mathrm{b}}$	$r_{\mathrm{a}}(1) - c^1_{\mathrm{a}}$
	b	$r_{\mathrm{b}}(2) - c^1_{\mathrm{b}}$	$r_{\mathrm{b}}(2) - c^1_{\mathrm{b}}$
	ab	$r_{\mathrm{b}}(2) - c^1_{\mathrm{b}}$	$r_{\mathrm{a}}(1) + r_{\mathrm{b}}(2) - c^2_{\mathrm{ab}}$
ab	$\emptyset$	$r_{\mathrm{a}}(1) + r_{\mathrm{b}}(1) - c^1_{\mathrm{ab}}$	0
	a	$r_{\mathrm{a}}(2) + r_{\mathrm{b}}(1) - c^1_{\mathrm{ab}}$	$r_{\mathrm{a}}(2) - c^2_{\mathrm{a}}$
	b	$r_{\mathrm{a}}(1) + r_{\mathrm{b}}(2) - c^1_{\mathrm{ab}}$	$r_{\mathrm{b}}(2) - c^2_{\mathrm{b}}$
	ab	$r_{\mathrm{a}}(2) + r_{\mathrm{b}}(2) - c^1_{\mathrm{ab}}$	$r_{\mathrm{a}}(2) + r_{\mathrm{b}}(2) - c^2_{\mathrm{ab}}$

Table 3: Generalized payoffs

As an example, with $r_1(1) = r_2(1) = 10, r_1(2) = r_2(2) = 4, c^1_{\mathrm{a}} = c^2_{\mathrm{b}} = 5, c^2_{\mathrm{a}} = c^1_{\mathrm{b}} = 6, c^1_{\mathrm{ab}} = c^2_{\mathrm{ab}} = 7$, we get the values in table 4.

For these values, the strategy of making only the "weaker" product (a for vendor 2, b for vendor 1) is dominated by making both products. The pure Nash equilibria are (a, b) and $(\mathrm{ab}, \mathrm{ab})$.

For other values of the variables, the set of equilibria is different. The critical conditions are:

$(\emptyset, \emptyset), (\emptyset, \mathrm{a}), (\emptyset, \mathrm{b}), (\mathrm{a}, \emptyset)$, and $(\mathrm{b}, \emptyset)$ can never be equilibria, since they leave monopolies to be taken, and monopolies are always attractive. $(\mathrm{ab}, \mathrm{a})$ cannot be equilibrium since vendor 2 would rather exit (equation 18). The same applies to $(\mathrm{b}, \mathrm{ab})$.

The remaining strategy profiles can become equilibria under the following conditions ($\neg$C1.1 stands for "not C1.1"):

Vendor 1	Vendor 2	Π_1	Π_2
$\emptyset$	$\emptyset$	0	0
	a	0	4
	b	0	5
	ab	0	13
a	$\emptyset$	5	0
	a	-1	-2
	b	5	5
	ab	-1	7
b	$\emptyset$	4	0
	a	4	4
	b	-2	-1
	ab	-2	7
ab	$\emptyset$	13	0
	a	7	-2
	b	7	-1
	ab	1	1

Table 4: Example payoffs

C1.1	$r_b(2) - c_b^1 + e^1 > 0$	Expanding into weaker product with
C1.2	$r_a(2) - c_a^2 + e^2 > 0$	competition is worthwhile
C2.1	$r_a(2) - c_a^1 + e^1 > 0$	Expanding into stronger product with
C2.2	$r_b(2) - c_b^2 + e^2 > 0$	competition is worthwhile
C3.1	$r_a(2) - c_a^1 > 0$	Producing stronger product with competition
C3.2	$r_b(2) - c_b^2 > 0$	has positive payoff. Implies 2.x
C4.1	$r_a(2) + r_b(2) - c_{ab}^1 > 0$	Making both products with competition has
C4.2	$r_a(2) + r_b(2) - c_{ab}^2 > 0$	positive payoff. Equals sum of C1.x and C3.x

Table 5: Critical conditions

- (a, b), if ¬C1.1 and ¬C1.2;
- (b, a), if ¬C2.1 and ¬C2.1;
- $(\emptyset, ab)$, if C3.1 and C4.1;
- $(ab, \emptyset)$, if C3.2 and C4.2;
- (a, ab), if C1.1, ¬C1.2, and C3.1;
- (ab, b), if C1.2, ¬C1.1, and C3.2;
- (ab, ab), if C1.1, C1.2, C4.1, and C4.2.

For an overview of the different regions, let's consider cases with a combined symmetry between vendors and products:

$$c_{\mathrm{a}}^1 = c_{\mathrm{b}}^2 = \overline{c} - h \tag{23}$$

$$c_{\mathrm{b}}^1 = c_{\mathrm{a}}^2 = \overline{c} + h \tag{24}$$

$$c_{\mathrm{ab}}^1 = c_{\mathrm{ab}}^2 = 2\overline{c} - e \tag{25}$$

$$r_{\mathrm{a}}(2) = r_{\mathrm{b}}(2) \tag{26}$$

$$r_{\mathrm{a}}(1) = r_{\mathrm{b}}(1) \tag{27}$$

This leaves a parameter space of 5 dimensions instead of the previous 10. $r_m(1)$ is irrelevant for the critical conditions, it only enters into the fixed assumptions above. Using $r_m(2)$ as a scaling factor leaves $\overline{c}$, h and e as relevant parameters. $\overline{c}$ and e will be influenced by the decision of the vendors to cooperate in building a shared component; h is not necessarily influenced.

Figure 6 shows a section through this space with a fixed cost level $\overline{c} > r(2)$, and variable cost asymmetry h and economies of scope e. Because of the combined symmetry, condition C1.1 and C1.2 are identical, as well as C2.1 and C2.2, and so on. The plane is divided into 5 regions with different sets of equilibria. Some results are intuitively plausible: E.g. (b,a) is an equilibrium when cost asymmetry is low; (ab,ab) is an equilibrium when economies of scope are high. The plane is bounded by the line $e = \overline{c} - h$ on the upper right corner, beyond which condition 15 on page 26 would be violated.

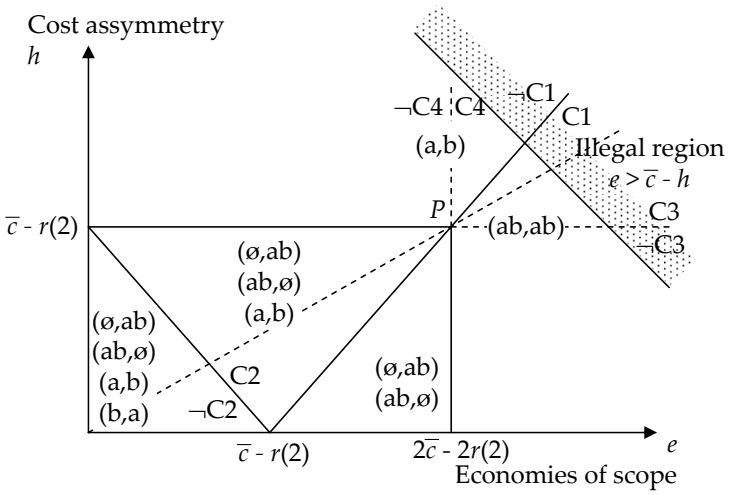

Figure 6: Section of parameter space with fixed cost level $\overline{c} > r(2)$

Figure 6 keeps its general shape with variations in $\overline{c}$ (and, of course, $r(2)$). As $\overline{c}$ grows (relative to $r(2)$), the rectangle between P and the origin becomes larger and, with it, the inscribed triangle. The boundary of the

illegal region also moves to the right, but only half as fast as P. At $c > \frac{3}{2}r(2)$, P lies in the illegal region. When $\bar{c}$ decreases, the rectangle shrinks, and at $c \leq r(2)$ it disappears, leaving only the two regions $\{(\mathrm{a},\mathrm{b})\}$ and $\{(\mathrm{ab},\mathrm{ab})\}$.

Within the rectangle, multiple equilibria are possible, and the question is: Which is most likely or preferred by the vendors? In any case, each product is produced by only one vendor, therefore there is no difference in revenue. The total cost in the case of $(\emptyset, \mathrm{ab})$ is $2\bar{c} - e$, and $2\bar{c} - 2h$ in the case of (a,b). Therefore, above the dotted line ($2h = e$), (a,b) is cheaper overall and leads to higher social welfare, and below this line $(\emptyset, \mathrm{ab})$ or $(\mathrm{ab}, \emptyset)$ are cheaper and hence desirable.

The equilibria (ab,b) and (a,ab) are not possible in the case of combined symmetry and hence do not appear in the diagram. If vendor 1 had a fixed, slight cost advantage, the diagram would be modified in such a way that the line representing C1 would split into a line C1.1 slightly below, and a line C1.2 slightly above it. In the space between these lines, above and to the right of p (more precisely above a line C3.2 which would lie slightly above the depicted C3), the unique equilibrium would be (ab,b). Likewise, (a,ab) would be possible if vendor 2 had a slight cost advantage.

Above the C2 boundary $h = \bar{c} - r(2) - e$, (b,a) is not only impossible, but the "weak product" strategies (b for vendor 1, a for vendor 2) are also dominated by the strategy ab.

Within the triangle formed by the C2 line, the dotted line from the origin through R ($h = e$) and the vertical line at $e = \bar{c} - r(2)$, there exists also a Nash equilibrium in mixed strategies:

$$\sigma_1(\emptyset) = \sigma_2(\emptyset) = \tfrac{1}{r(1)-r(2)}(\bar{c} - r(2) - e) \tag{28}$$

$$\sigma_1(\mathrm{a}) = \sigma_2(\mathrm{b}) = \tfrac{1}{r(1)-r(2)}(e - h) \tag{29}$$

$$\sigma_1(\mathrm{b}) = \sigma_2(\mathrm{a}) = 0 \tag{30}$$

$$\sigma_1(\mathrm{ab}) = \sigma_2(\mathrm{ab}) = \tfrac{1}{r(1)-r(2)}(r(1) - \bar{c} + h) \tag{31}$$

The payoff for both vendors in this mixed equilibrium is 0.

Figure 7 shows a section through the same space with a fixed cost asymmetry $h < \frac{1}{2}r(2)$. One can identify the same regions and boundaries. With rising h, Line C1 descends, while the boundary of the illegal region and C2 move upward. When $h > \frac{1}{2}r(2)$, C1 lies completely within the illegal region, and therefore only three regions remain. Left of the vertical line at $e = 2h$, (a,b) leads to higher social welfare and total vendor profit than $(\emptyset, \mathrm{ab})$. The mixed Nash Equilibrium exists in the strip between the C2 boundary and the line through S parallel to it, and to the right of the $e = h$ line.

In this figure it is quite easy to see the possible transitions from one set of equilibria to another when a shared component is introduced. A shared

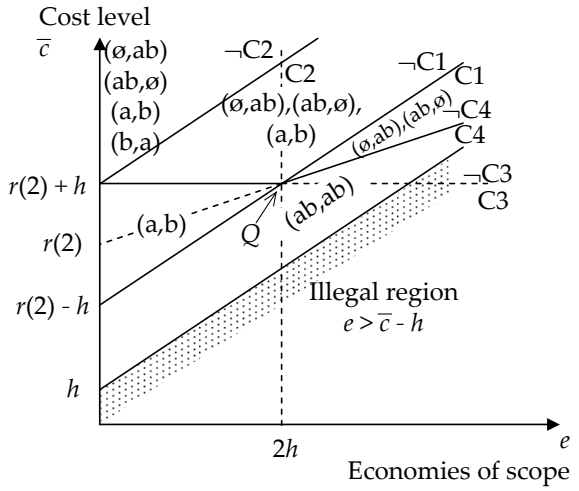

Figure 7: Section of parameter space with fixed cost asymmetry $h < \frac{1}{2}r(2)$

component has the effect of lowering $\overline{c}$ by some amount, and lowering e by the same amount or a little more (cf. equation 22 on 27). h can be expected to hardly change. From any point representing the situation without a shared component, we get to the situation with a shared component by following an arrow pointing southwest to west-southwest. Figure 8 shows the possible region changes.

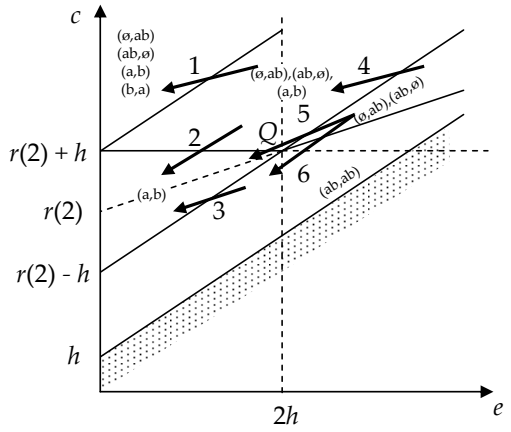

Figure 8: Possible region changes through introduction of a component

Of course, in many cases, there is no region change, i.e. the same structural equilibria apply to both the case with a shared component as without

a shared component. Without a change in the structural equilibrium, building a shared component is worthwhile overall if and only if both vendors produce products.

A change following arrow 1 in figure 8 does not have much influence, as the three most plausible equilibria are available "before" (without the shared component) and "after" (with the shared component). Arrow 2 is more interesting. Without the shared component, vendors face the danger of "falling into" equilibria $(\emptyset, \mathrm{ab})$ or $(\mathrm{ab}, \emptyset)$, which, when we are left of Q, is less desirable than (a,b). With the shared component, vendors can both exclude this inefficiency and realize the component-related economy. Arrow 2 could in principle also originate from the right of Q. Then, the economy of scope of a single vendor is in competition with the component-related economy of scope (plus the specialization effect).

Arrow 3 is an even more interesting case for vendors, and shows how even vendors who compete in both products can have an incentive to cooperate in components — under the assumption that anti-trust law allows the latter, but prohibits explicitly dividing up the market. Arrow 4 as shown is similar to arrow 1 — the added equilibrium (a,b) is not very plausible when one is far to the right of Q. But if arrow 4 were longer, (a,b) could become more likely.

Arrow 5 only applies for fairly restricted parameter choices. Again the vendor-related economy of scope are played out against component-related economy of scope plus specialization. Arrow 6 leads from a monopoly situation to vendors competing in both products, which would be good for buyers, but so unattractive to vendors that they will in most cases refrain from sharing a component.

We are now ready to discuss the first, cooperative stage of the game. The simplest case is when there is no region change between the situation with and without a component, and only one equilibrium in the region. This is the case with regions $\{(\mathrm{ab}, \mathrm{ab})\}$ and $\{(\mathrm{a}, \mathrm{b})\}$. In both cases, sharing a component is worthwhile and will therefore be achieved by rational negotiators. The economy to be shared is $2\Delta_z c_{\mathrm{ab}} - c_z = 4\Delta_z \bar{c} - 2\Delta_z e - c_z$ for $\{(\mathrm{ab}, \mathrm{ab})\}$, and $\Delta_z c_{\mathrm{a}}^1 + \Delta_z c_{\mathrm{b}}^2 - c_z = 2\Delta_z \bar{c} - c_z$ (assuming $\Delta_z h = 0$). Any distribution of this economy between the vendors is in the core of the game, and a fair solution may be achieved by splitting it 50/50.

If there is no region change, but multiple equilibria, the contribution of the shared component depends on the individual equilibrium. For a typical case, take $\{(\mathrm{a}, \mathrm{b}), (\emptyset, \mathrm{ab}), (\mathrm{ab}, \emptyset)\}$. Sharing a component is worthwhile overall if players choose (a,b), but will reduce total payoff if one of the other equilibria is reached. If a player agrees to share the component, this could therefore be interpreted as a weak signal that he is ready to divide markets with the other player, and therefore to avoid a costly fight. Of course, players still cannot make a binding commitment regarding the second phase. The signaling effect is not achieved when the component is offered by a

third party to both vendors.

If there is a region change, the signaling effect is stronger, and may also be achieved by a third component vendor. For example, in the case of Arrow 2, sharing the component makes it very clear that players intend to play (a,b) in the second stage. Arrow 3 is a clear case where we can always expect rational vendors to engage in sharing a component, because both vendors payoffs are higher in that case.

References

[Art89] W Brian Arthur. Competing technologies, increasing returns, and lock-in by historical events. *Economic Journal*, 99(394):116–31, March 1989.

[BF94] Stanley M. Besen and Joseph Farrell. Choosing how to compete: Strategies and tactics in standardization. *J of Economic Perspectives*, 8(2):117–131, 1994.

[CG92] Jeffrey Church and Neil Gandal. Network effects, software provision, and standardization. *Journal of Industrial Economics*, 40(1):85–103, March 1992.

[CG93] Jeffrey Church and Neil Gandal. Complementary network externalities and technological adoption. *International Journal of Industrial Organization*, 11(2):239–260, June 1993.

[Cou38] A. Cournot. *Researches into the Mathematical Principles of the Theory of Wealth*. Kelley (1971 Reprint), New York, 1838.

[Eco89] Nicholas Economides. Desirability of compatibility in the absence of network externalities. *American Economic Review*, 79(5):1165–81, December 1989.

[Eco91] N. Economides. Compatibility and the creation of shared networks. In M.E. Guerrin-Calvert and S. S. Wildman, editors, *Electronic Services Networks: A Business and Public Policy Challenge*. Praeger, 1991.

[ES92] Nicholas Economides and Steven C Salop. Competition and integration among complements, and network market structure. *Journal of Industrial Economics*, 40(1):105–23, March 1992.

[EW95] Nicholas Economides and Lawrence J. White. Access and interconnection pricing: How efficient is the "efficient component pricing rule"? *Antitrust Bulletin*, XL(3):557–579, Fall 1995. Available from: `http://www.stern.nyu.edu/networks/site.html`.

[Fie05] Roy T. Fielding. Software architectures in an open source world (presentation). In *ICSE*, 2005.

[FS86] Joseph Farrell and Garth Saloner. Standardization and variety. *Economics Letters*, 20(1):71–74, 1986.

[FS92] Joseph Farrell and Garth Saloner. Converters, compatibility, and the control of interfaces. *Journal of Industrial Economics*, 40(1):9–35, March 1992.

[GC02] Annabelle Gawer and Michael A. Cusumano. *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Harvard Business School Press, 2002.

[GKKS04] Bernhard Gröne, Andreas Knöpfel, Rudolf Kugel, and Oliver Schmidt. The apache modeling project, October 2004. Available from: `http://apache.hpi.uni-potsdam.de/`.

[GT79] J.J. Gabszewicz and J.F. Thisse. Price competition, quality and income disparities. *Journal of Economic Theory*, 20(3):340–359, 1979.

[Hil97] CWL Hill. Establishing a standard: Competitive strategy and technological standards in winner-take-all industries. *Academy of Management Executive*, 11(2):7–25, 1997.

[Hot29] H. Hotelling. Stability in competition. *The Economic Journal*, 39(153):41–57, 1929.

[KS85] Michael L Katz and Carl Shapiro. Network externalities, competition, and compatibility. *American Economic Review*, 75(3):424–40, June 1985.

[MR88] Carmen Matutes and P. Regibeau. Mix and match: Product compatibility without network externalities. *Rand Journal of Economics*, 1988.

[MR92] C. Matutes and P. Regibeau. Compatibility and bundling of complementary goods in a duopoly. *The Journal of Industrial Economics*, 40(1):37–54, 1992.

[Por80] Michael E. Porter. *Competitive Strategy*. The Free Press, 1980.

[Tha] Robert S. Thau. Shambhala api notes. Available from: `http://httpd.apache.org/dev/API.html`.

[Tha96] Robert Thau. Design considerations for the apache server api. In *5th International World Wide Web Conference, Paris*, 1996.

.