# NOKIA ON THE SLOPE: THE FAILURE OF A HYBRID OPEN/CLOSED SOURCE MODEL

<table>
<thead>
<tr>
<th>Journal:</th>
<th>The International Journal of Entrepreneurship and Innovation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID:</td>
<td>IEI-17-0099</td>
</tr>
<tr>
<td>Manuscript Type:</td>
<td>Teaching Case Study</td>
</tr>
<tr>
<td>Keywords:</td>
<td>open source, knowledge integration, hybrid innovation model, Nokia</td>
</tr>
<tr>
<td>Abstract:</td>
<td>This case study explores the origins of Nokia’s decline in the mobile technology market, as an unsuccessful attempt to introduce an open source strategy into the business. Nokia created a hybrid model, which codified conflicting principles taken from closed and open mode of collaboration. A series of implementation problems resulted in Nokia struggling to attract open source partners, growing issues with managing in-house staff and ultimately failing to develop a new mobile operating system fast enough to stay competitive. Key learning outcomes: At the competition of the case study students will: • understand complexity of open innovation implementation when paradigmatic differences between businesses and/or partners are not resolved;</td>
</tr>
</tbody>
</table>
- be able to critically evaluate a hybrid knowledge innovation model and how it might create difficulties at the operational level
- identify challenges of management in relation to open source software developers,
- analyse strategies which corporations may employ to successfully benefit from the open source software (and similar) movement and how organizational hypocrisy can lead to failure of open innovation projects.

Intended use: This case study is best suited for post-graduate or executive courses.
NOKIA ON THE SLOPE: THE FAILURE OF A HYBRID OPEN/CLOSED SOURCE MODEL

Background

Open innovation has become a new paradigm for understanding industrial progress. It works on the premise that widely distributed knowledge leads to innovation, which occurs at the interspaces between a diverse range of groups and organizations (Swan & Scarbrough, 2005). Primarily to advance their technology and systems, businesses need to make use of both internal and external ideas (Chesbrough, 2003). Open innovation models enable customers and users to contribute their knowledge to a company’s R&D processes. Many companies now understand the benefits of this and have incorporated such models to strengthen their creativity and profits (Weber, 2004). Open source software (OSS) is the common field in which open innovation takes place. However, we must remember that open source software is a community-based model of development, not a business model per se, and its existence precedes the concept of open innovation.

Extant studies suggest that the open innovation models offer many opportunities, although it also bear risks. Although it can be argued that open innovation is inevitable, it remains a challenge for business organizations seeking to benefit from external knowledge sources but still operating like closed systems. The change in the business model requires not only adapted structures and systems, but also redefining of a company’s role in those structures and systems (Ciesielska, 2010). The open paradigm also challenges the traditional proprietary business approach because it is ruled by a different intellectual
property regimes and populated by variety of participants with different ideologies. But above all, a fully open R&D process makes a company more vulnerable to its competitors. It also risks that their competitive advantage may be easier to copy. However, somewhat ironically, the whole body of open innovation research offers little in-depth analysis of the organizational and legitimacy problems involved in creating a hybrid open source/corporate structure. Ruuska et al. (2009) argue that any knowledge integration requires crossing not only temporal, spatial, and task boundaries but also authority, social and identity boundaries. The former ones may be less visible and harder to overcome. Generation of knowledge is a collective activity and as such requires substantial levels of coordination between heterogeneous agents and common coherent institutional background in order to succeed (Antonelli, 2006): ‘Knowledge does not spill freely and automatically in the atmosphere: dedicated efforts are necessary to create the institutional context into which external knowledge can be acquired and to reduce its uncontrolled leakage.’ (Antonelli 2006: 234) Literature identify employees, R&D personnel (Schroll and Mild, 2011), non-R&D personnel (Robertson et al., 2012) and managers (Jones, 2006) as important gatekeeper for absorbing new knowledge. Laviolette et al. (2016) report that both non-research and development personnel can play a crucial role in inbound open innovation.

In their seminal work, Meyer and Rowan (1977) noted that individuals in organizations tend to get involved in symbolic actions that aim to decouple from formal structures. Decoupling means that organizations tend to adopt various formal solutions in response to a range of institutional pressures, but do not necessarily enact those structures. When organizations have to deal with incommensurable institutional or stakeholders’ pressures, they often start to couple their activities, talks, and decisions in odd manners (Brunsson, 2003). The concept of organizational hypocrisy explains how organizations struggling for legitimacy and satisfying interests of different parties talk and decide about important...
issues, showing their involvement in one direction while in practice acting in the opposite direction. This implies that, if a company makes many statements and reveals future plans, actual executions may be far away (or even opposite) from those declarations. However, the organizational hypocrisy strategy only works if people assume that talks are leading to coherent actions and it is not revealed that the organization is decoupling its talks from activities (Brunsson, 2003). Swan and Scarbrough (2005), in their study on the politics of networked innovation, highlight that the power of the system may constitute barriers to innovation and make knowledge integration efforts less effective. Enberg et al. (2006) argue that little effort has been devoted to developing alternatives to rationalistic project management concepts of how and with what mechanisms knowledge integration processes are utilized in the context of uncertainty and political pressures.

The case

In 2013 Nokia sold their mobile branch to Microsoft for $7.17 billion. After several years of outstanding success and rising profits, the company had already failed in its transition to smart phones and a customer-focused business model (Doz & Kosonen, 2008). But the problems started in the early 2000s when the company seemed to lose its strategic sensitivity. This is possibly due to a great success during the 1990’s, which made its leaders underestimate rising competitive threats and a new phase of technological progress in the mobile industry (Ciesielska, 2010). Although it is true that Nokia did not stop working on incremental changes to its operating system, Symbian, its major investment and hopes were put into the open source Maemo project. As accurately noted by the open innovation expert Henry Chesbrough, Nokia’s approach to innovation should have embraced radical changes. ‘This company that achieved so much with its product design in the 1990s must develop an entirely new set of innovation skills in order to create,
develop, and manage a platform— an ecosystem of other companies that build their offerings on top of Nokia’s.’ (Chesbrough, 2011:5), but this has not fulfilled. Instead this case shows that interaction between organizations that share an open environment often results in emerging paradoxes (Munoz & Lu, 2011).

From Symbian to Maemo

During the ten years leading up to the Microsoft takeover in 2013, Nokia’s Internet tablet/smart phone product announcements and releases were entangled with organizational changes in the company (See Fig 1). Nokia started to work on a multi-media device quite early. The development process was initiated in 2002 with the Nokia 7700, which was based on Symbian. Symbian was an old proprietary operating system which Nokia had used for years. However, it soon became obvious that the current platform had its limitations and would potentially be unsuitable as an operating system for new types of embedded mobile devices, such as smart phones. New, innovative solutions were clearly needed, but Nokia didn’t seem to have any ideas, or a business partner that could provide an answer.

At that time, there was a very small number of Nokia engineers who had links to the OSS world and this was enough to start a quasi-hobbyist project to develop a Linux-based mobile operating system. Linux is an example of free open source software and, since its development in 1991, it has become the main operating system for the world’s super computers. Initially, Nokia’s management did not take the idea seriously. In fact, this attitude continued until the engineering team came up with the first actual product prototype: Tablet N770.

Tablet N770 offered an Opera internet browser, internet radio access and a media player. The device and its operating system, Maemo, were officially announced in 2005 at the
Linux World Fair and Conference in New York City. It was at this point that Nokia’s management declared that the Maemo project was an initial step in creating an open source product for broadband and internet services; they also stated that the company would now be coordinating and regularly launching new versions of the software.

At this stage all Nokia had was an internet-enabled prototype device, but this gave hope for maintaining their competitive position in the future. With the right investment and attitude, Nokia had a chance to create a device which incorporated new and exciting smart phone technology. However, Nokia’s development process was relatively slow and other big players in the market (primarily Google and Apple) had already been working intensively on the idea of smart phones packed with internet browsing and multimedia utilities.

The second generation of tablet, N800, was rolled out in January 2007. In October that same year, the third-generation, N810. The operating system was upgraded to Maemo 3, then to version 4; the devices were additionally equipped with an integrated camera and a Skype/VoIP application. Unlike previous tablets, the N810 release had a side-slider with a full QWERTY-keyboard, together with a touch screen, a Mozilla internet browser, GoogleTalk and GPS navigation.

However, until 2009 and the N900 model, the internet tablet was a product without a SIM card designed for internet browsing via a Wireless Local Area Network (WLAN). The N900 was the first embedded system that successfully combined the mobility and size of a cellular phone and, to a certain extent, the usability of a laptop. Unlike a regular phone, it was programmable according to the user’s needs and they could develop, upgrade and install software as desired. In 2009, Nokia was working on the Maemo 5 system, which also incorporated Qt libraries.

The following section describes Nokia’ hybrid model, taking into account key organizational principles related to intellectual property rights, knowledge distribution,
membership, incentives, authority and coordination of the project, and the dominant mode of communication.

**Nokia’s Hybrid Model**

Nokia became involved with open source software in two distinct ways. Firstly, it contributed to well-established projects, such as the Linux kernel or GNOME. GNOME is a desktop environment for GNU/Linux and UNIX-type operating systems and it is used for managing applications. The code from these established projects became the basis for creating an operation system on Nokia’s internet tablets. The second way Nokia got involved in OSS was by starting its own open source community. The Maemo.org website, and its community, was aimed at producing free applications for Nokia devices. The code from Maemo.org was supposed to be free of charge (both for Nokia and its customers) via an application store.

However, not all components of the new software were open sourced. Some of it came from commercial vendors, or was developed internally by the company and released as a closed module. The mixed structure in which Nokia hoped to bring innovation was an open source/business hybrid. This hybrid was constructed around a mixture of organizing principles which merged the open source and commercial mode of software development.

**Knowledge Integration and Distribution**

The operating system for internet tablets, Maemo (later MeeGo), was composed primarily of free and open source software. Nokia’s vision (or perhaps propagated visualization) of Maemo was its centrality, in the sense that it took bits and pieces from different OSS projects and compiled them into the operating system via the cooperation of the Nokia and Maemo communities (Maemo.org website). Approximately 25 percent of the packages on
the Maemo platform were taken directly from open source software upstream projects; 50 percent of the packages originated in OS upstream projects, but were modified internally by Nokia; the remaining 25 percent of the packages constituted proprietary components (Jaaksi, 2007). Thus, approximately 75 percent of the operating system for Nokia tablets had external inputs. Nokia was involved in those projects at various levels of engagement. This participation in the upstream OSS projects enabled the access and transfer of necessary knowledge to develop Maemo OS. A volatile balance seemed to exist between different interests regarding knowledge sharing. On the one hand, Nokia started to understand the advantages of following and directly contributing to upstream processes. Initially, they tried to fork the projects and develop them internally, but they then had to subcontract companies to post factum reintegrate the code. The critical point in this process was the translation of open source logics on the corporate grounds, and explaining to the managers where the gains came from. According to Ari Jaaksi, Nokia’s vice president, its work with Maemo taught Nokia to avoid ‘forking’ the code. To benefit fully from open source processes, a company should make an effort not to deviate significantly from its upstream components; if they are working on their own version of the source code repository, too many differences make it difficult to incorporate software upgrades.

**Intellectual Property**

Nokia was a corporation that, for many years, focused on the closed R&D model. Not long before it started integrating OSS, its research centre explicitly stated that its mission was to produce more patents. After Nokia became involved in OSS development in 2008, it had to officially reformulate its strategy and (at least within OSS operations) switch goals from increasing the number of useful patents to expanding its collaborative networks.
At the same time, Nokia stated on wiki.maemo.org that the OSS licensing model is preferred by the company. But despite this, Nokia went on to execute a highly confusing approach, both internally and externally. It applied a ‘novelty rule’ for its engineers, one which stated that any code potentially constituting a ‘competitive advantage’ should not be open sourced unless agreed with the management. In fact, this attempt at a competitive advantage differentiation is perhaps the most important factor in this whole argument: The nature of most open source licensing makes it impossible to patent any top-up solutions. If the ‘novelty’ rule were taken to extremes, many crucial upgrades might not have been returned to the upstream projects, which is against community spirit. It seems that the differentiation between clear and novelty was not only uncertain to begin with, but that the definition was also at the discretion of various levels of management within the company, which further confused an already unstable hybrid business model. The reality of Nokia’s activities not only contradicted some of its strategy presentations, but they were also incoherent in themselves. This did not go unnoticed. On many occasions during conferences, meetings, blog posts and private conversations, OSS developers unofficially complained about Nokia creating double standards.

**Membership**

Typical open source software projects are voluntary and gather together independent developers, corporate employees, open source firm programmers, translators and users. The main issue facing Nokia’s Maemo-related network was not how to restrict membership, as it is in more commercially-oriented project, but rather the difficulty in engaging capable participants who were interested in developing projects for Nokia (preferably for free).

Nokia became involved in the open source movement by employing developers through a number of arrangements, including: Work contracts; subcontracting; supporting companies
via task-oriented contracts; and sponsoring talented individuals in non-contractual, ‘favour-for-a-favour’ relationships. Most developers involved in open source operations at Nokia were previously engaged in major OSS projects. From the company side, it seemed to be a conscious strategy to recruit people with proven expertise, in order to acquire necessary knowledge and skills; as well as advancing Nokia’s reputation and position in OSS communities.

However, the recruitment process wasn’t easy and on several occasions the brightest people within OSS weren’t willing to enter into any formal agreements with Nokia. Consequently, the corporation needed to try and interest a developer in a particular bug or problem and hope that he or she would work out a solution. So by employing, subcontracting, or sponsoring people in this way to do the open source-related jobs, Nokia pushed its own projects forward and enabled developers to make a living from their hobby-projects. However, many open source developers didn’t want to pursue long-term career plans with Nokia and sustaining loyalty became a major issue.

**Coordination**

Authority and control in the open source world is a difficult thing to maintain. Nokia tried to keep an influence over many OSS organizations, usually by sponsoring and participating in foundation boards. This ensured that the corporation kept a voice in what was developed and influenced its development further, at least formally. However, the decisions made by core developers were, to a large extent, entirely independent. Because of this the upstream OSS project stayed beyond Nokia’s direct control.

Nokia, as all other participants in this process, could download updates and submit code to the main repository (subject to acceptance from the core development group). Thus, on upstream open source grounds, Nokia had full power over its own actions and its own
submissions. However, it could never have been sure which alignments would be made between its own code and the upstream one. This seemed to be a serious concern for top management and, in this context, Nokia’s decision to buy Qt library was clearly an effort to gain full control over one of the projects and its submissions.

At the same time Maemo’s highly-controlled environment actively discouraged many talented developers. As well as this, Nokia also learnt that attracting such developers, and gaining their full commitment, didn’t just require their trust in Nokia’s ability to run the project effectively. Working with them primarily required trust in the company’s declaration to support the open source movement. The company’s indecisiveness about their sharing policy, and their difficulty in recruiting permanent staff and relocating them to Finland, resulted in a high staff turnover and lowered commitment. HRM became even more challenging as a result.

**Communication**

Despite Nokia’s declared openness, OSS developers faced problems with communication which came into direct conflict with open source principles. For instance, the corporation’s internet security policy created some burden on OSS development. Since open source work is done primarily online and, as a result, dual solutions were implemented. For instance, there was a separate, unofficial infrastructure available for the OSS contributors, while the rest of the employees were using the standard, well-protected network.

As well as internet access, Nokia was also extremely careful about what was allowed to be publicly said about its plans and R&D activities. Officially, almost no decisions were made public until the last moment, just before the release of a product or a piece of software. In practice, a lot of plans and activities became *secret de Polichinelle*, as they were already distributed among trusted informal networks and never released to the rest of the
company. On many occasions, Nokia’s open source system developers admitted revealing more details of their internal work than what was formally permitted. This happened in various ways, such as by hinting at external collaborators or allowing outsiders to Nokia to test unreleased pieces of software. And although such behaviour went against corporate policy, there was not a single case when a developer was sanctioned for it. Middle management usually understood that this is how open source collaboration worked and so another inconsistency became common practice.

TEACHING NOTE

1. Case synopsis

This paper presents Nokia’s journey from closed to open knowledge integration model. It shows that initially easy start with involvement in open source communities and being able to make use of open source software is not enough to succeed with open innovation. Several key management issues which contributed to Nokia’s failure are highlighted: an inability to engage external contributors and gain trust in open source software communities, the implementation of an unclear, hybrid business model with open collaboration being used within the company’s core competitive advantage area, and HRM-related problems related to managing open source style of work.

2. Learning outcomes

At the competition of the case study students will:

- understand complexity of open innovation implementation when paradigmatic differences between businesses and/or partners are not resolved;
- be able to critically evaluate a hybrid knowledge innovation model and how it might create difficulties at the operational level
• identify challenges of management in relation to open source software developers,

• analyse strategies which corporations may employ to successfully benefit from the open source software (and similar) movement and how organizational hypocrisy can lead to failure of open innovation projects.

3. Discussion questions & analysis

a. What are the potential benefits and risks with implementation of open innovation paradigm?

Extant studies suggest that the open innovation model offers many opportunities for businesses. By introducing more diversity, linking variety of people and companies, it encourages organizational change and ‘the loose coupling between the innovation process of the firm and its business model invites close examination of this coupling’ (Chesbrough, 2006a:33).

However, the open innovation model also bears many risks. A fully open R&D process makes a company more vulnerable to its rivals by making their competitive advantage easier to copy. Although the strategic perspective argues that open innovation is inevitable, it remains a challenge for business organizations seeking to benefit from external knowledge sources but still operating like closed systems. The change in the business model requires not only adapted structures and systems, but also the redefining of a company’s role in those structures and systems (Ciesielska & Westenholz, 2016).

b. Why did new Nokia knowledge integration model become problematic over time?

It is clear that Nokia needed a better balance between closed and open knowledge integration models. Totally unrestricted knowledge sharing would have made all R&D processes available to direct competitors. On the other hand, Nokia learnt that sharing-only OSS collaboration is beneficial for a business. Nokia was taking the source code from
the OSS projects and changing two thirds of it in-house, for use on the internet tablet. This
seemed not to be cost or time effective. Although Nokia claimed that it was making efforts
to ensure that the modifications were accepted into the main projects, insider’s
perspectives reveal that the internal policy was driven by the competitive advantage rule.
Such a policy resulted in submitting only minor repairs of bugs to open source projects and
thus contradicted the rules of the open source system (Ciesielska & Westenholz, 2016).

The Nokia case accumulated problems related to knowledge integration from both closed
and open innovation models (Ulhøi, 2004). This happened because Nokia had to deal with
internal functional divisions, while an external OSS community promoted an ethic that was
ideologically contradictory and hard to control. Technical details and other types of
knowledge translations were judged by their commercial value and kept in secret. But from
the OSS perspective knowledge sharing is a basis for collaboration. Nokia hybrid model
and its problems are summarised in table 1.

c. How did Nokia’s management deal with employment and motivation of
Open Source Software developers?

From the internal knowledge integration perspective, developers often complained about
constant re-organization. At one point, they had to adjust to new structures and processes
every several months. Since open source operations tasks were connected only to
software development, they had to cooperate with the hardware designers. Not only does
this type of knowledge divide both departments, but the company was also divided
ideologically. The case shows that not enough effort was put into proper integration and
co-ordination, both at ideological and practical level.

Other problematic issues included strategic management decisions and their practical
consequences. Some employees believed that strategic decisions were sometimes taken
without real consideration for technical issues and at meetings developers were rarely
listened to. Eventually Nokia’s managers found that developing a hybrid structure closely linked to, and dependent on, an exterior collaboration network, required them to manage the coexistence of conflicting values. However, that knowledge came to them too late, as they constantly failed to maintain and span boundaries between public and private, open and closed and contractual worker versus professional developer (Ciesielska, 2010). Other research also shows the in the absence of scripts for how to deal with the tensions between logics, there is ambiguity around choice of goals, target population, or management principles (Battilana & Dorado, 2010).

d. Can you identify key strategies that could have prevented Nokia from failing?

i. Building Long-term Relationships with OSS Communities

Because open innovation is dependent on external partners and contributors, the company must spend time and effort engaging with selected open source software communities. Those communities usually differ in their logic and governance from business organisations (Jemielaniak, 2014), which may encourage organisational hypocrisy rather that actual strategic and operational change. Building trust and being honest about a company’s involvement and ensuring the company brings value to the greater project, are essential to facilitate collaboration. It is important to have a coherent strategy of how the company will get involved and sustain their contributions to the open source project, without risking a loss of business or being considered as a community customer (Ciesielska & Iskoujina, 2012; Ciesielska & Petersen, 2013).

ii. Rethinking Competitive Advantage

Implementation of a hybrid business model that introduces open innovation into the company offers many new R&D opportunities. However, deep involvement in an open source project requires both inbound and outbound open innovation. This means that the
company cannot only be a community customer, but is expected to effectively contribute to the project. Working in the open makes the company vulnerable to its competitors. Therefore, it is crucial that the company rethinks their competitive advantage and makes sure the open collaboration doesn’t threaten their market position (Chesbrough, 2006b).

iii. Tackling Internal difficulties of Implementation

All strategic change will cause implementation problems and switching from a closed innovation model to a hybrid one is not an exception. In this case, the particular difficulty comes from the need to mix conflicting ideas about how internal organization and HRM should work. The case shows that the hybrid organization concept may seem easy to implement in theory, but in practice there are a range of ideological and commitment problems to proposed solutions. By tackling them early enough, a company can prevent turf wars and workforce distress, thus saving money and potentially saving the business entirely. More about hybrid originsations and responses to contradictions within them can be found in Battilana & Dorado (2010), Jay (2013) and Battilana & Lee (2014).

References


Figure 1: Nokia's internet tablet development line; source: the author, developed from Ciesielska, 2010:38.
<table>
<thead>
<tr>
<th>Organizing principles</th>
<th>Nokia’s business–OS hybrid</th>
<th>Problems with implementation: Enacting conflicting organizational principles</th>
</tr>
</thead>
</table>
| Knowledge integration and distribution                    | 1. 3 sources of code: OSS, subcontractors, and Nokia internally. Some software components are closed (proprietary) while others use OSS licensing. Main burdens for knowledge creation:  
   2. Boundary objects: the code repository, documentation, financial documentation, business plans, prototypes, etc.  
   3. Most key internal solutions are kept in-house, key external solutions are incorporated; the rule of “competitive advantage” | - Internally: lack of specialized knowledge of Open Source Software development; need to coordinate activities among departments and managers; traditionally proprietary mind-set background  
   - Externally: securing submission primarily solving Nokia’s, but not wider OSS community’s problems; corporate plans vs. fragmented OSS process |
| Intellectual Property Ownership                          | Knowledge is quasi-public, different licensing types used                                   | Internally: unclear, introducing confusion                                                                                                                                                   |
| Membership restriction and Incentives                    | 1. Membership is open (externals), although there is a core of selected employees and subcontractors;  
   2. Salaries, fees, sponsoring, creating learning possibilities, creating, building career possibilities, and assessing candidates for job offers | Willingness to work for Nokia, either formally or voluntary depends on the trust in their expertise, and trust in the company’s declaration to support open source project. Difficulty in recruiting permanent staff and relocate them to Finland. Loyalty problems. |
| Authority and coordination                               | Problematic (peer/company review and control), the ‘connecting people approach’  
   Encouraging self-organization within the limits) | Clash of open source mentality with corporate strategy. Identity problems. Inconsistency in policies slow the development. |
| Dominant mode of communication                           | Technology-mediated, face-to-face interactions at meetings/conferences (usually more often than once a year, networking, PR) | Open source developers striving to work in the open, while corporate policy is to retaining Intellectual Property. |