Frans-Willem Duijnhouwer

Chris Widdows

AUGUST 2003

Open Source Maturity Model

In order to be able to determine if an open source product is suitable for an organisation, Capgemini has developed the Open Source Maturity Model. This model allows you to determine if or which open source product is suitable using just seven clear steps. Not only a good way to keep interesting but immature products away from your business, but also an useful tool to objectify the discussion on applying Open Source in the workplace. Given the current strong political interest for Open Source, having a professional approach to open source products is an asset.

Capgemini

✉ fduijnho@capgemini.nl

☎ +31 30 6896651

(Office)

2

THE USEFULNESS OF A MATURITY MODEL

It isn't wise to be clever to quickly

It is easy to be satisfied about exchanging commercial product XYZ for open source product OSABC. Meanwhile dark clouds are gathering over the organisation, the storm of problems caused by unfounded use of a new product is definitely approaching.

The Capgemini Open Source Maturity Model allows a correct application of open source products. In a short timeframe, using some simple steps it shows what is right and what isn't.

OPEN SOURCE IN A CORPORATE SETTING

The concept of Open Source raises lots of questions, in particular when the application of Open Source within a corporation is discussed. Open Source products have many advantages, but are easily categorised as 'cheap' and 'nerdy' software. This usually prevents them from properly being considered during a product selection phase. Which is a pity, because a proper review of these products would show that there are situations in which open source products can be used effectively.

Views of openness

Open Source doesn't use one common set of licensing terms, but what binds them all is the thought that intellectual access shouldn't be restricted. Some are very protective and demand that any future development should remain as unrestricted as the source. The GNU GPL license[1] is a good example. Others are less restrictive, the BSD Linux license[2] for example. The term 'open' or 'free' refers to the intellectual knowledge within the software product. It doesn't refer to the economic aspect that deals with the use or acquisition of that knowledge.

Advantages

One of the Open Software advantages is the fact that an organisation owns the product components, in sharp contrast with commercial software that increasingly favors a subscription model. This affects the continuity of the production process, for example by having to follow the mandatory 'upgrade' policies that cause an enormous 'lock-in'.

The absence of commercial interest in Open Source software builders is seen by many as an important advantage, even if it only seems to promise a nicer / kinder view of the world.

Another advantage is that the cost of the software is usually significantly lower than with a similar commercial product. The openness (due to availability of the source code) is also big advantage for corporations looking for modifications to the product. Knowledgeable parties can add to the original source code, allowing the product to remain coherent and still offer new functionality.

[1] GNU GPL is the www.GNU.org General Public License
[2] BSD is Berkely Systems Development

Open source can cause confusion

The openness in technology is not the same as clarity for the potential user audience. For instance the confusing version numbers, which rarely indicate any sense of usability (some very usable products have yet to reach version 1.0) make it hard to evaluate the product potential. Another reoccurring issue is complexity of matching the feature set to business requirements. Capgemini recognised this challenge and has looked for ways to allow an organisation to approach Open Source in a professional and business-like manner. This has resulted in the Open Source Maturity Model (OSMM), which provides a systematic approach for evaluating, implementing Open Source products within a corporate environment.

Open Source Maturity Model

The OSMM describes how an Open Source product should be assessed to ensure that the product meets the IT challenges companies face today. The OSMM accomplishes this by linking an extensive product analysis with a thorough review of the company and its IT issues.

Capgemini's knowledge and experience serves a the 'glue' that links these two parts together to product a clear, independent and reliable overview of the use of a Open Source product within a specific company.

The OSMM is a part of the process used by Capgemini to produce an independent advice. One in which Open Source is on a level pegging with regular commercial offerings.

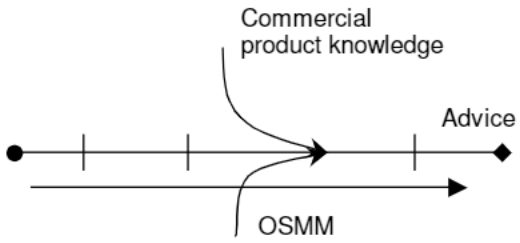


Figure 1 The OSMM as an integral part of the Capgemini advice process

The OSMM brings Open Source expertise to the selection and advisement process and forms a natural addition to the process already used by Capgemini. In close cooperation with the customer the OSMM allows Capgemini to:

1. **Determine** the maturity of a Open Source product,
2. **Access** a Open Source product's match to the business requirements,
3. **Compare** Open Source products with commercial alternatives,
4. **Show** the importance of an Open Source Partner (OSP).

After evaluating a range of products that might qualify as a solution for the formulated business requirements the suitability of those products (which is a combination of product maturity and feature match) becomes clear. The customer can then make an informed decision about the use of Open Source software in the workplace.

4 PRODUCT SELECTION

Currently the number of Open Source projects is strongly increasing. The developer communities are growing both in size and geographic spread. Another frequent occurrence is that an Open Source project is ended due the emergence of an alternative that is accepted by the user community. This set of dynamics must be dealt with, but how? Do we incorporate products that show enormous potential but still fall short into the product portfolio or not? How is a product that is 'hot' but still a little 'not' valued / compared?

To answer these questions we need to compare Open Source projects in an objective manner.

One way of looking into the future is examining the quality of the release management. In the commercial software world it is quite common to view a 1.0 release with some trepidation. But in the world of Open Source it is quite common to find beta versions being used for production. So the infamous 1.0 release usually requires a more positive evaluation when assessing Open Source than its commercial counterpart.

Features in time

A very important part of the process is the product's feature set. A product that has features that need to be implemented will be much less suitable in the workplace when compared to one that already has all the required features. Especially with software it pays to formulate policy based on what is available instead of on what is promised. This implies that any product cannot participate in the selection process purely based on expectation.

If a product shows potential and in time will deliver a richer feature set than its nearest competitors, it could be wise to postpone the final comparison for a certain period. Such a re-evaluation would only be carried out if the product is properly managed and the expectations are clearly above average, in short; is this product mature enough to wait for? This allows the product to evolve into a top contender, which results in a more accurate final advice. Any postponement is subject to being able to wait until a second measurement can be made. At which time all the products will be re-measured to provide an accurate measurement.

OPEN SOURCE PRODUCT INDICATORS

How to compare apples and oranges?

The review of an Open Source product is different from a review of a commercial product. Whereas a commercial product strives to protect the intellectual property, an Open Source product is focussed on sharing that same property.

A brief overview of some of the differences clearly shows why the key comparison indicators are so different.

DIFFERENCES BETWEEN OPEN SOURCE AND COMMERCIAL PRODUCTS

Open Source products are freely available and naturally that is not the case for commercial products. Generally the users of a commercial product do not receive the source code of the product, but users of an Open Source product expect to receive the source code. Some of other differences between these types are presented in the table below.

	Commercial	Open Source
Supplier	A company	A community
Product development	Driven by corporate economics	Driven by product functionality
Developers	Limited numbers with product knowledge, all paid for the supplier	Varies from a small to very large group of developers. Often permanently employed, sponsored or volunteers.
Stability	New trends are incorporated quickly if there is a commercial incentive.	New ICT developments are incorporated into the product if this benefits the users.
Users	Commonly not organised, every user maintains contact with the supplier independently.	Users participate in virtual communities and discuss among themselves and with the developers about the product and future developments.

There are many more differences between Open Source software and commercial software. But the table already shows that there is genuine need to select Open Source products based on a different set of indicators from the ones used for commercial products.

Open Source product review

The review of the product is conducted using a number of objective and measurable facts. This is because the emphasis in this stage is on how the product came to be and the success it has in obtaining a market share. Those units of measure are termed the **product indicators**.

Besides the product indicators it is also important to assess which aspects of the product are relevant within the specific context of the review and how the product scores on those aspects. Factors like maintenance, training facilities, connectivity to existing infrastructure and interoperability to other products. These indicators are strongly driven by the needs of the

customer and are not pre-defined by Capgemini. The OSMM describes which steps Capgemini takes to determine these indicators, how to score on these indicators and how the selection a product is achieved. These indicators are called **application indicators**.

PRODUCT INDICATORS IN DETAIL

Product indicators are grouped into 4 different groups:

- *Product*
- *Integration*
- *Use*
- *Acceptance*

Each of these groups consists of a number of indicators, which together form the product score. The group *Product* focuses on the 'internals' of the product, things like the development and stability of the developer group or the purpose of the product. The group *Integration* measures the options to link the product to other products or infrastructure. In addition it is also a measure for the product's modularity. The *Use* group tells us something about the way in which the user is supported in everyday use of the product. For instance by reviewing the number of support options made available to the user. The *Acceptance* group is all about the way the product is received in the user community, as this is largely indicative of the product's ability to grow and become a prominent product.

The manner in which Capgemini applies the indicator to an Open Source product is described for each product indicator. Application indicators can only be scored according to information provided by the customer during interviews. Without a customer there can be no *application* of an Open Source product.

Age (Product)

The longer a product remains under active development, the smaller the chance becomes that the developers suddenly stop. For all Open Source initiatives the first year is the largest hurdle. Commonly the initiative is halted due to lack of response (lots of work, no glory) or that the group is too small to sustain the workload that the product generates. As long as there's no financial compensation for all this effort the group must attract new developers or seek a product sponsor. Either of these will allow the group to sustain the development effort.

Selling points (Product)

Products with a clearly defined selling point more easily gain market share. Take Qmail (a MTA[3]) for example. Despite sendmail being the accepted standard, it still managed to gain a sizeable market share and create an active community. The reason: Qmail was developed because of difficulty experienced when trying to configure sendmail to be a secure MTA. Qmail's selling point; it addressed an important weakness of sendmail. But if we look at all the different Open Source GUI toolkits we see a different picture. Despite a large number of high quality toolkits being available, new ones keep popping up again and again. The newcomers have no clear selling points and usually don't last very long.

Developer community (Product)

The saying 'many hands make light work' certainly holds true when dealing with an Open Source product. Changes to people's ambitions and personal life frequently result in that person moving on to do other things. The greater the group of active developers the less chance that the product development stalls. A large group also requires that the group must organise to continue to effectively work together. Group organisation is one of the driving forces behind an effective community.

Human hierarchies (Product)

Projects with one all-controlling leader tend to last just a short time. Projects that delegate control to other active members (usually dividing the project into separate areas of attention, allowing the original 'captains' to explore newer avenues) have matured more. It not only allows the project to grow whilst maintaining the stable' version, it gradually increases the supporting community

Licensing (Product)

Open Source products can choose from several different licenses. The choice made tells us something about the way in which the intended users are approached. Some are very restrictive, so restrictive in fact that they become the subject of the discussion if the product can still be considered Open Source under such a license. Some companies try to find ways in attracting large numbers of non-paid developers (something Netscape tried for some time), allowing the company to lower the development costs. Others offer several licenses, even catering for commercial variants (MySQL does this).

Collaboration with other products (Integration)

As the product gains acceptance within the target audience the call for the ability to work with other products is heard more often. Usually the request to be able to 'script' certain aspects of the product's functionality is the first feature that points to interoperability. The next step is the incorporation of more structural changes, like using PAM (Plugable Authentication Module, a generic interface to allow interaction between a product and a separate authentication system) for example. Collaboration with other products is therefore the result of change requests that have been accepted by the development team. So it isn't just the product that is collaborating.

[3] MTA is a Mail Transfer Agent

Modularity (Integration)

As an Open Source product gains more market share, others could become more interested in parts of the product's functionality. This allows the developers to develop a more flexible licensing scheme (protecting the core, but allowing fewer restrictions on other parts), which could even allow commercial developments to hook up to parts of the system. By splitting the product into several modules commercial interest can be attracted without sacrificing the Open Source principle. This is what happened with Xfree (XFree86), the X-server used by most Linux distributions. Previously video card manufacturers where required to give out all the card's inner details to get Xfree to work with their card. This was changed to allow (closed source) binary drivers to hook up with the rest of Xfree. The users gained access too much more recent and powerful hardware and video card manufacturers were now more involved in supporting Xfree. The opposite also happens, commercial products are offered in a trimmed down version to users as Open Source, while the full product remains commercial (closed software).

Standards (Integration)

In the world of commercial software setting your own standard is still seen as a way to protect the investment in the product. For Open Source products adhering to widespread and accepted standards is extremely important. By only supporting standards that are common in just a couple of environments can adversely influence the acceptance of a product. This depends a lot on the product itself, for instance a product can connect to a database using a direct connection (limited to certain databases), ODBC (standard in use in Unix/Linux and windows) or OLEDB (standard only used in windows).

Support (Use)

With some products the support is obtained by mailing the single developer. Others maintain a discussion group (or even groups), but only a couple of regulars respond to request. A few maintain very active discussion groups in which a large number of members will offer support. Some products will even guarantee support if you pay them. The manner in which support is given or offered says a lot about the way the development group takes its users seriously.

Ease of deployment (Use)

If a product becomes so popular that even independent parties start to offer training courses, it is almost a certainty that it has become a mature product; the very least it has become a very popular product. More commonly seen with Open Source is that active users start writing task specific papers (HOWTO's). These HOWTO's allow new users to accomplish the desired functionality without having to master all of the product capabilities. HOWTO's cover all aspects of product usage, not only how to set-up for a particular application, but also how to maintain the product. Existence of documentation detailing day-to-day maintenance is an indication of maturity.

User community (Acceptance)

Some products generate hardly any noise; some have several busy discussion groups. When an Open Source product is well received it is common to witness an outburst of user requests, suggestions and problem reports. The discussion group quickly fills up with large numbers of messages, so the developers must expand and start to manage this huge flow of feedback. This could be described as an Open Source project's puberty. An active community is not to be underestimated. When Sybase stopped all development on Watcom C++ (a developers tool) the community rallied and negotiated an open source option. Today it lives on as Open Watcom C++.

Market penetration (Acceptance)

The installed base tells us something about the importance of the product within the intended users. A product with a large installed base (Apache for example) provides additional stimulus to form communities. Users will want to voice new requirements, discuss problems, and therefore require a platform to do so. Some users may have clear ideas on how to advance the products; others will appreciate the possibility to communicate directly with members of the development team. A lager installed base indicates a more mature product.

PRODUCT INDICATORS

Capgemini uses the following indicators to access the maturity of Open Source products.

Indicator	Immature	Mature
Product		
Age	The project has just started. The stability of the developers group and need for the product are unclear.	The project is been active for some time. The project stability and need for the project are no longer issues.
Licensing	Not fully described or clearly unsuitable for the product.	One of the 'standard' licences (www.opensource.org/licenses/) Offers clear motives for choosing the license type, which is supported by the user community. Often allows both commercial and Open Source variants to co-exist.
Human hierarchies	Original founder is lead developer and solely responsible. Development depends on a single person.	Large community, multiple leaders who coordinate. Separation of development and maintenance.
Selling points [4]	Only enthusiasm.	Commercial issues like security or maintainability.
Developer community	Small tight knit group.	Very active developers community, several hand-offs have taken place. Documented procedures to becoming a member.
Integration		
Modularity	No modules, still one single product. Functionality is offered on a 'take all or nothing' basis.	Product has been separated into smaller pieces of functionality. Users can select which parts are required. Allows tailoring of the product to a particular situation.
Collaboration with other products	Not in focus yet. Product development is still firmly centred on core functionality.	Product is close to completion. Attention is shifting to linking the product to other products.
Use		
Standards	Uses propriety protocols or uses dead end technologies.	Uses current accepted protocols and models. Deals with issues surrounding standards, integration etc.
Support	Just within the own community and then only provided by a small minority within that community.	Besides community support, professional support can be purchased. A SLA can be negotiated. The community itself is active and questions draw responses from a wide section of the community.

[4] Open Source projects usually start because of some frustration with current offerings. This results in projects that 'wanted to try it themselves' or 'want to become main product because the current ones are good enough'.

Indicator	Immature	Mature
Acceptance		
Ease of deployment	Little to no training facilities or courses. Documentation is poor, particularly with regard to maintenance.	Training or courses available. In addition to well written documentation lots of HowTo[5]'s of users detailing particular situations. Within the group knowledge about maintaining the product is readily available.
User Community	Small group, possible with a high proportion of 'lurkers[6]'.	Large group that often has divided itself into sub-groups. Each group has a specific focus. Traffic in general is best described as high-volume.
Market penetration	Few references, just local promotion. Low exposure is the reason the product isn't wildly known.	Often mentioned by others (for example Gartner, ZDNet, Netcraft, IDC). Multiple cases of successful implementation across a range of companies. Well known.

Capgemini has reviewed many Open Source products and has extensive experience using these indicators and how they are applied to Open Source products. Because of this, Capgemini now has a portfolio containing various Open Source product reviews to which new reviews are being added. Existing reviews are updated on a regular basis, ensuring accuracy of the reviews contained in the portfolio. The use of a consistent approach ensures an independent, reproducible and professional result.

Because Capgemini is willing to commit to an Open Source product recommendation and ensure that the customer is provided with sustainable and maintainable solution the maturity measurement is a vital part of the process. There is a clear relation between the level of maturity of a product and the risk that the project development will come to an end. Due to the unique nature of Open Source, even a product that is no longer under active development can still be successfully deployed. Support can then be provided by a smaller number of developers.

[5] A document detailing on how to use the product for a particular task or in particular situation. Usually written by a 'fellow' user.
[6] People that are interested in using the product but show little inclination to contribute to it.

A COMPARISON

The table below shows the scoring criteria for the indicators.

Product indicator	Score: 1	Score: 3	Score: 5
Product			
Age	< 2 months	1-2 years	> 3 years
Licensing	Unclear / unknown	Clear (GPL, LGPL). OSI approved.	Multiple options catering to the user's requirements.
Human hierarchies	Individual	Club	Organisation
Selling points	Individual enthusiasm	Group enthusiasm	Business motives
Developer community	Not clear how to become a member	Becoming a member is possible, but requires initiative	Clearly documented procedures. Mentions skill and roles available. Use of group development tools is required.
Integration			
Modularity	One large non-extendable application	Limited expansion, for instance by scripting	Defines a plug-in architecture to allow other parties to incorporate modules.
Collaboration with other products	Stand alone	Partial interaction (possibly by using certain protocols).	Incorporates accepted standards (for example: PAM)
Standards	Propriety	Outdated	Latest industry standards.
Use			
Support	By developers	By product users and developers	By independent support companies, product users, developers
Ease of deployment	Little to no documentation. Maintenance documentation is non-existant.	Product reference only. Documentation focusses on the product, not on its usage.	Lots of HowTo's for a wide range of situations, third party training. Users with extensive maintenance experience. Focus on the deployment of the product.
Acceptance			
User Community	Perhaps a mailing list	One or two moderated groups about all aspects of the product.	Multiple moderated groups, each one dealing with distinct aspects of the product. Users are active, forming unofficial groups and thinking up new initiatives
Market penetration	Unknown	A viable alternative	Market leader

Because some indicators cannot be measured in a purely numerical sense; the score is determined by a panel of Capgemini experts who have demonstrated knowledge of Open

Source and have worked with similar products. Because of the use of this 'panel' the score cannot be set by one single person, this ensures that the score is objective.

If an indicator isn't applicable for a particular product, the score is set to 3. Because Capgemini always uses a threshold value of 3+, setting the score to three ensures that this indicator does not affect the outcome in a positive or negative way. The determination of maturity is therefore always controlled by the indicators receiving a below or above 3 score. This maturity threshold is a cut-off point; below this cut-off maturity is not enough to qualify for further selection.

APPLICATION INDICATORS

An Open Source product can't (just as any other product) be introduced into a working environment based solely on a measurement of it strengths and weaknesses. To properly assess product one must also take into account several environmental aspects and naturally the present and future demands of the user. The OSMM of Capgemini takes these factors into account by defining the following application indicators:

- **Usability** – The intended user audience, the experience of that group.
- **Interfacing** – Required connectivity, which standards are applicable. How does this fit into the organisation?
- **Performance** – The expected load and processing capability. The performance demands that must be met.
- **Reliability** – What level of availability should the product deliver?
- **Security** – What security measures are required, what restrictions are imposed onto the product.
- **Proven technology** – Does the product use technology that has proven itself in daily production?
- **Vendor independence** – What level of commitment between supplier and user does the product demand?
- **Platform independence** – Is the product available for particular ICT environments only, or does the product allow a wide range of platforms.
- **Support** – What level of support is required.
- **Reporting** – What reporting facilities are required.
- **Administration** – Does the product allow the use of existing maintenance tools, the demands for operational management.
- **Advice** – Does the client require validation / recommendation by independent parties, if so, what is required.
- **Training** – Required training and facilities.
- **Staffing** – Is product expertise bought, taught or hired.
- **Implementation** – Which implementation scenario is preferred?

The data of all these indicators is collected, user requirements are determined so that it becomes possible to access if the product is suitable or not. In addition the importance the

client attaches to each of these indicators is also recorded, scoring on a scale of 1 to 5, 1 being 'not important' and 5 being 'extremely important'.

For a number of the indicators the trend analysis is important. This is why the data is recorded for three periods, the present, near future (between now and + 6 months) and distant future (within 2 years). If it is impossible to determine the future value, or if that value is of no importance the value of the previous period is used. Finally, the importance of the period is also recorded.

THE MATCH

All of this data is combined to the 'final' score that indicates the suitability of the product for the given demands. Determining one single score allows an easy comparison between candidate products.

Key indicators

Besides the cold yes/no that the use of single score causes it is important to highlight trends and key aspects. This enables the customer to recognise indicators that could 'turn' the score. This is especially important when a customer reassesses the importance of certain indicators. By recognising the key indicators an organisation can quickly determine the impact of change without having to do a complete reassessment of all the products.

Currently Capgemini has an extensive collection of Open Source product profiles. These reusable 'components' contribute to a quick and professional evaluation of possibility to use Open Source within an organisation.

14

THE MODEL

Capturing awkward data in a model

One of the fundamental properties of Open Source products is that they exhibit an irregular improvement 'roadmap'. A primary reason for this phenomenon is that improvements are mostly requested, the user community has a requirement that is addressed. With commercial software a new version also has an economic motive; it generates income.

This irregular development path that is native to Open Source products causes some products to improve at a very high rate, whereas the products competitors could show a much more relaxed approach. The developer and user community are responsible for these differences. Because of this variance updating the maturity model on a regular basis is imperative.

OPEN SOURCE MATURITY MODEL

The product indicators that were mentioned earlier form the basis of the model. Using these indicators the maturity of Open Source products can be determined. Capgemini consultants with extensive Open Source product experience have determined the product indicators. The application indicators are prioritised together with the customer.

Product indicators receive a score valued between one and five. One is poor, five is excellent and 3 is average. All the scores are summed to produce a product score. The value of this score tells us something about the general maturity of the product. Capgemini also maintains a minimum score that serves as a threshold for products that represent such a high level of immaturity that they shouldn't be considered for professional application. The total per group (product, integration, use and acceptance) says something about the maturity of the product for that particular group.

Besides the product indicators the Open Source products are also scored for requirements that relate to the manner in which the application must ultimately perform; the application indicators. Both the customer experts and Capgemini consultants score these indicators. During one or more interviews the customer fills a questionnaire about his/her preferences in relation to a number of products, any of which could potentially meet the requirements. This is done using the product indicators.

In addition the customer and Capgemini consultants will determine which indicators must absolutely score above average. These indicators are the key indicators.

The scoring of the application indicators by the customer is discussed with a Capgemini expert. This expert has also completed a scoring, in which his/her knowledge of the situation is applied. By comparing notes it is possible to eliminate the grey areas and determine the final scoring. By combining that score with the product maturity score a total score in which both the product, the application of the product and specific customer demands are all combined; a total score for each product.

A mandatory part of the OSMM process is the feedback phase. This is to evaluate the effectiveness of the indicators. As the workplace changes, so should the OSMM. Evaluation ensures that the indicators remain finely tuned and effective. As Capgemini continuously

evaluates the product indicators by researching and participating in the various Open Source communities the OSMM as a whole always remains up-to-date.

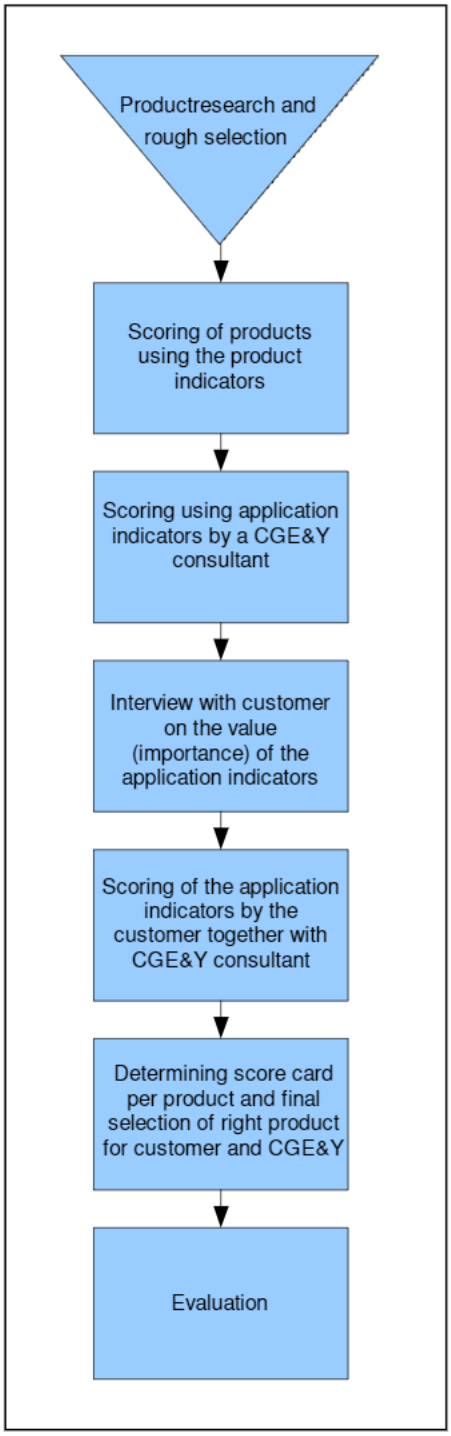
AN EXAMPLE

To show how the OSMM calculates an example is presented below.

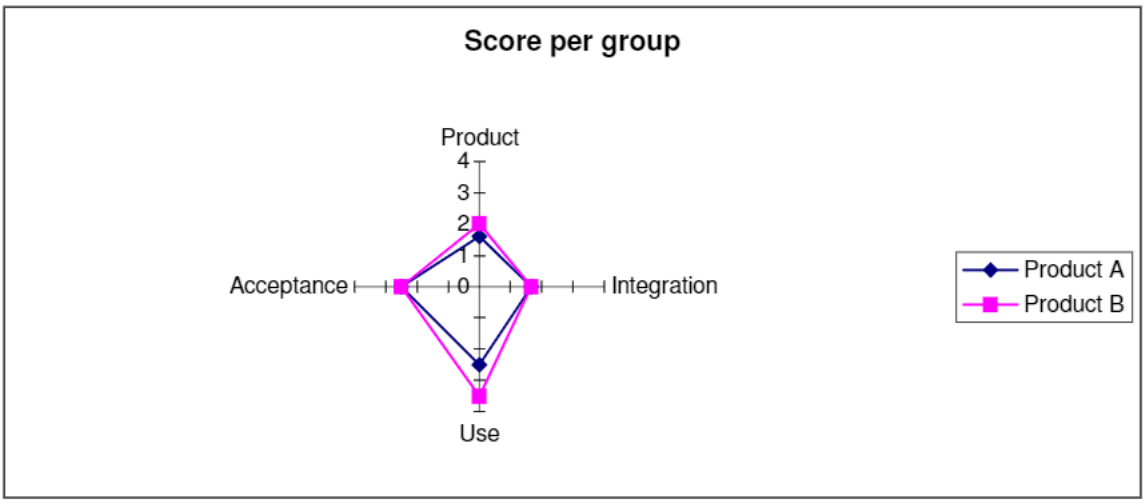
In this example we'll compare two non-existing products, A and B. First the product indicators are discussed, followed by the application indicators.

Product indicators

Indicator	Product A	Product B
Product		
Age	1	3
Licensing	2	2
Human hierarchies	1	2
Selling points	2	1
Developer community	2	2
Integration		
Modularity	1	1
Collaboration with other products	2	2
Standards	2	2
Use		
Support	2	3
Ease of deployment	3	4
Acceptance		
User community	2	3
Market penetration	3	2
TOTAL	**23**	**27**



The table allows us to determine a score for each group of indicators. This allows us to compare products on a group basis, for this example we can draw this graph:



It becomes clear that both products score well on *use,* but both products fall short when it comes to integration.

In more detail, this example shows that according to the product indicators product A is less mature than product B. This means that Capgemini is less keen to use product A. Capgemini has a threshold, below which the use of the product in a professional setting is not advised. If a product fails to meet the minimum maturity criteria this will be explained in detail, together with the risks and a 'maturity prediction' (i.e. an assessment of the way in which the maturity of this product will develop in the next 6 months based on the knowledge Capgemini has). Even though the match on application indicators has a higher priority than the product indicator, if the score on application indicators does not decide for one single product Capgemini will prefer the more mature option. Likewise, even if the application match is near perfect, if the product fails to meet the maturity threshold the OSMM will not select that product as the preferred product.

To match the requirements of the customer to the strengths and weaknesses of the product the customer and a Capgemini consultant have scored the product according to the application indicators.

Application indicators

		Product A		Product B	
Indicator	**Priority (P)**	Score (S)	**PxS**	Score (S)	**PxS**
Usability	5	5	**25**	1	**5**
Interfacing	4	4	**16**	2	**8**
Performance	3	3	**9**	3	**9**
Reliability	4	2	**8**	4	**16**
Security	4	1	**4**	5	**20**
Proven technology	5	5	**25**	1	**5**
Vendor independence	2	4	**8**	2	**4**
Platform independence	2	3	**6**	3	**6**
Support	3	2	**6**	4	**12**
Reporting	1	1	**1**	5	**5**
Administration	3	5	**15**	1	**3**
Advice	2	4	**8**	2	**4**
Training	2	3	**6**	3	**6**
Staffing	5	2	**10**	4	**20**
Implementation	1	1	**1**	5	**5**
TOTAL			**148**		**128**

This shows that product B doesn't match the customer requirements as well as product A.

The score is mainly determined by the weight factor (P), which is determined for each indicator and the score that the customer and Capgemini consultant give for each indicator. Because both products meet the maturity threshold, the OSMM determines that product A is best suited for this particular situation. Even though product B is more mature, the application indicators show that this maturity doesn't offer any additional value to the customer in this case.

When an indicator that is crucial to the customer receives a poor score; this could mean that the product will not be considered for further selection. For each selection certain show-stop indicators must be determined. If a product is scored below average on those particular indicators, it is rejected.

When a comparison is made between Open Source products and commercial products the same application indicators are used. The product indicators will be different. Essentially it isn't the community but a company that is judged on the same ability: to provide a product over a certain period of time. But unlike with an Open Source product the option to provide this from our own resources is not available.

EVALUATION, THE SECRET OF SUCCES.

As with anything that measures, calibration is important. The IT market changes strongly from year to year; with new ways to use IT in the workplace being discovered continuously. The IT products also change, just as the idea of using Open Source product wasn't really considered to be realistic until recently. Changes in the political, economic climate and recent changes to the licensing policy of some companies have focussed the attention on Open Source products as a viable alternative. Independent companies have conducted extensive research and concluded that for quite a few IT solutions Open Source offers a worthwhile alternative. The OSMM provides a way to assess this alternative on its own merits.

To continue to profit from the advantages that the OSMM offers the continuous calibration of the indicators is essential. Capgemini consultants constantly check the product indicators to ensure they remain accurate. This principle is also applied to the product profiles of previously scored products. When a product changes (such as a new release, new functionality, different licensing etc..) Capgemini re-scores the product. Capgemini Open Source trend watchers continuously monitor the validity of the indicators and maintain the accuracy of these indicators.

Our business consultants monitor the application indicators, but ultimately these indicators apply to our customers. This is why the OSMM process explicitly requires that our consultants use the feedback of our customers as a way to maintain the accuracy of these indicators.

The basis for the evaluation is put in place when the OSMM is actually used. The discussions between our customers and consultants offer ample opportunity to present new and unique insight to the evaluation process. Our consultants are required to check is any of these new ideas warrants a permanent change to the OSMM. In our normal dealings with our customers we constantly check to see if the ideas and decisions used in the evaluation are still valid.

By embedding these evaluations into the OSMM process they become a powerful instrument to ensure the highest level of quality. Not just now, but also in the future.