RC23990 (W0606-122) June 26, 2006
Computer Science

IBM Research Report

Metrics-based Management of Software Product Portfolios

Sunita Chulani, P. Santhanam
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Brent Hodges
IBM Software Quality

Kelley Blacksten
IBM Software Group

Metrics-based Management of Software Product Portfolios

Sunita Chulani & P. Santhanam
IBM T. J. Watson Research Center
({sunita, pasanth}@us.ibm.com)

Brent Hodges
IBM Software Quality
(hodgesb@us.ibm.com)

Kelley Blacksten
IBM Software Group
(kelleyb@us.ibm.com)

Abstract:

This paper targets organizations that develop and support a broad portfolio of software products or applications. We discuss a metrics framework to assist in managing such a portfolio, with implications to the financial, developmental, and support aspects. This framework is based on a heuristic grouping of products into three categories: Emerging, Growth and Stable and a set of product characteristics as captured by the observed trends in market share, functionality, development cost, support cost, quality, customer cost of ownership, etc. We focus mainly on one characteristic, namely quality (as represented by software defects and customer reported/field problems) and its impact on service costs. We analyze actual data for 27 software products to illustrate the basic ideas and highlight some lessons learned. We discuss various business scenarios for exploiting this framework that managers can use.

Keywords*:*

Software product portfolio, software product and process metrics, software quality management, software measurement frameworks, knowledge management.

1. Introduction:

Commercial software product vendors (e.g. Microsoft, IBM, Oracle, etc.) develop and manage a large portfolio of software products spanning operating systems, middleware, firmware, applications, etc. Industrial organizations (e.g. banking, telecommunications, automotive, etc.) also create and manage large number of custom business applications. At the portfolio level, managers at all these companies face an important problem: how do we manage the investments, revenues, quality and customer expectations across a wide variety of software products and applications? To this end, we have developed a heuristic framework that is based on a grouping of products, defined by a set of product characteristics as captured by the observed trends in market share, functionality, development cost, support cost, quality, customer cost of ownership, etc. This framework is similar to the technology adoption life cycle as described by Moore in [1]. There are different types of customers for technology products ranging from early adopters of the product (the technology enthusiasts and visionaries), early majority (the pragmatists) and conservatives and skeptics. Moore argues that this is because visionaries and enthusiasts have very different expectations from pragmatists. Jones presents a different view by differentiating amongst projects (vs. adopters of the technology) and groups them into leading, average and laggards [2]. He claims that laggards have poor quality whereas leaders have very good quality.

There has been considerable research focused on the development/maintenance of a single product and its different releases/versions. Well-established models such as COCOMO [3] and SLIM [4], address the area of in-process software metrics. Other authors have tried to incorporate a broader scope of quality metrics, including product attributes, into the software assessment tool kit. For instance, [5] provides details of the AS/400 development project in IBM Rochester, which won the Baldrige Quality award. Mendonca [6] did an extensive study using the Goal-Question-Metric approach on product attributes such as Capability, Usability, *etc.,* and provided ways of improving measurement of these attributes. None of these approaches targets the management of a portfolio of software products or applications. While some of the ideas presented here may be practiced informally by managers, we believe that giving a more formal structure to this practice can help the IT industry substantially.

This paper presents examples of metrics that are collected and available at IBM to assist the technical and business decision making process, in particular, metrics that are used to classify product portfolio based on the maturity of the products. Section 2 describes the heuristics based framework scheme. Section 3 presents the lessons learned and Section 4 presents the value and usage of the framework. Section 5 presents the conclusions and next steps.

In this paper, when we say ‘product’ we mean either a product or an application. A product includes the first release of software and any subsequent versions and releases.

2. Heuristics based Product Maturity Framework:

Software products have many characteristics which vary with time such as their functional requirements, quality attributes, and time to market. Therefore, they cannot be managed the same, otherwise certain adverse conditions resulting from poorly made decisions can affect a number of aspects of the business. One method of preventing this is to use a framework for managing a software portfolio based on product maturity and the characteristics exhibited by a product as it matures in the marketplace**.** The particular framework discussed in this paper has implications for financial, developmental, and support aspects of a software portfolio.

Product Maturity levels take into account the length of time that a software product is in the marketplace, its install base, and the planned investment in new functional requirements. Based on these characteristics, a product’s maturity level can be categorized into one of three categories of increasing maturity: ***Emerging, Growth and Stable*** [7]**.**

Emerging products have been in the market generally for less than two years, have a small install base with expectations of growth and have investment plans to grow market share. ***Growth*** products have been in the market typically for two to five years, with an install base that is expected to continue its growth trend and continued investment plans to grow market share. Stable products have been in the market for more than 5 years, have a strong install base that is maintained at a flat trend, and very little or no plans for substantial additional investment for new requirements.

There are several other distinguishing characteristics that differentiate among the product maturity categories as shown in table 1. Some characteristics (or metrics) are associated with the software owner/vendor organization and some with the customer. These metrics are described briefly in the next few paragraphs.

2.1 Software owner/vendor considerations:

	Product Maturity	Emerging	Growth	Stable
Software Owner/Vendor Considerations	**Market share**	Establish market presence	Grow market share	Lead and hold market share
	Time to market	Very critical	Moderately critical	Less critical
	Product Quality	Could be improved	Reasonably good	Good
	New functionality	Adding aggressively	Adding continuously	Adding slowly and only if absolutely needed
	Development Cost*	High	Medium	Low
	Service Cost*	High	Medium	Low
Customer Considerations	**Adoption profile**	Early adopters for absolutely new ground breaking technology	Early adopters of new but proven technology	Adopters of very stable and reliable technology
	Stability / Reliability requirements	Low	Medium	High
	Cost of ownership	High	Medium	Low

Table 1: Product Maturity Categories

* Development and service costs are relative to revenue (see section 2.1)

As shown in table 1, metrics such as market share, product quality and new functionality of the developed product along with their associated development and support expenses are specific to the development organization. Expenses are judged typically relative to the revenues generated by the products in the market. The factors which determine revenue are the number of licenses and the unit price which can depend on product features and market value of the product or business revenue generated by the custom applications. The market share for Emerging and Growth products continues to grow whereas Stable products lead and tend to hold the market share. Time to market (TTM) plays a more critical role for Emerging products to create new business opportunity. It is less important when the customer base is established for Stable products when quality considerations may have higher priority than TTM issues. The quality of Emerging products may not always be comparable to that of products which have matured in their market. It could also be true that Emerging products may not have quality as their focus. However, the quality typically continues to improve with the maturity of the product with Stable products generally having the best quality. The development cost relative to revenue increases as a function of time for Emerging products, flattens out for Growth products and decreases for Stable products. The management strategy may also determine development costs; for higher maturity products, the strategy may be to spend less on adding functionality relative to revenue with an aim to maintain high quality and a strong customer base. Thus, the development cost tends to start at its highest level, steadily decreasing over time, until, when a product reaches the Stable category, it is reduced to maintenance levels. The support costs follow almost the same trend as development costs - due to a growing market share, Emerging products have a growing trend of defects associated with them resulting in higher service costs compared to more mature products. On the other hand, the cost per problem for Stable products may be much higher compared to Emerging or Growth products. Also, for less mature products, market share tends to dominate support costs whereas for more mature products quality plays an important role in determining service costs.

2.2 Customer considerations:

As shown in table 1, metrics such as the adoption profile of the customer, stability and reliability requirements and cost of ownership differentiate amongst our customers for the different product maturity categories. Typically our early adopters of Emerging products have a higher tolerance level for products which do not yet exhibit the quality characteristics ranging from usability issues to product failures. Our new technology customers are more open to acquiring Growth products and have a nominal level of tolerance. Whereas our customers who require a reliable and stable environment choose our Stable products. Furthermore, the overall cost of ownership for Emerging and Growth products, since they often offer new ground breaking technology functionality is initially quite high and flattens out for Stable products. Also, the higher risk of using a lower maturity product adds to the cost of ownership.

2.3 Limitations of the current framework:

In this framework we assume a normal evolution of a product which is not influenced by unexpected or planned business conditions that result in declining market share, leading to loss of revenue or the termination of product. All the products discussed in this paper follow the normal evolution. Some examples of scenarios that are not dealt with in this paper are presented below.

Scenario 1: A product may experience declining market share due to unplanned business considerations. Even though the product meets customer expectations and is better than other products in the market, it may lose its market share due to software from open source or other competitor's advantages such as monopoly.

Scenario 2: A reasonably good product may lose its market share to another competitor product of improved quality and better functionality. In such a case, the product development team may have become complacent and created room for the competitor to catch up resulting in declining market share.

Scenario 3: Customers may prefer to use a legacy product as is, due to the fact that it is adequate for their need and the alternatives may be too expensive and unjustifiable. From the software vendor perspective, this scenario can result in revenue without too much of development or support expenses.

Scenario 4: An application owner may decide to replace the current application with another that is based on improved architecture, better usability, etc. to achieve a strategic business goal. For example, a monolithic legacy application may be replaced by an application based on service oriented architecture.

3. Lessons learned from the usage of framework

We had data for 27 products from our product portfolio grouped into the maturity categories defined in section 2. The products were middleware communication, messaging and application server products. The resulting grouping had 6 products classified as Emerging, 9 as Growth and the remaining 12 as Stable. For each product, we had quality data in terms of field problems and defects over time. If a field problem requires a change in code or documentation, a defect is created. Several studies have been done on analyzing such quality data [8, 9, 10, 11] but we did our analysis using the framework described in section 2.

Lesson #1: It is the trend of defects, not their volume that determines maturity of the product.

As can be seen from figure 1, the general trend for field problem receipts and defects over time for Emerging products is increasing suggesting that the product quality can be improved, (but may not

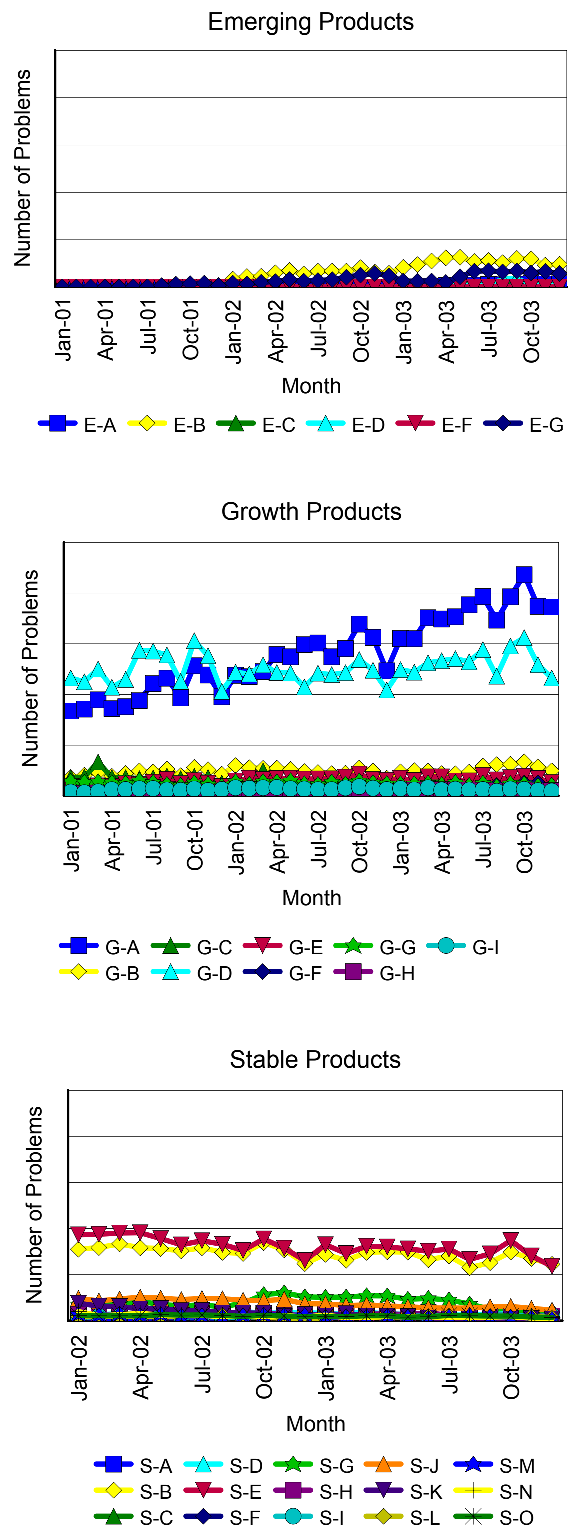

Figure 1: Trend in field problems

Note: Even though the ordinate is not labeled, we have kept the same scale for the three graphs above to facilitate easier comparison.

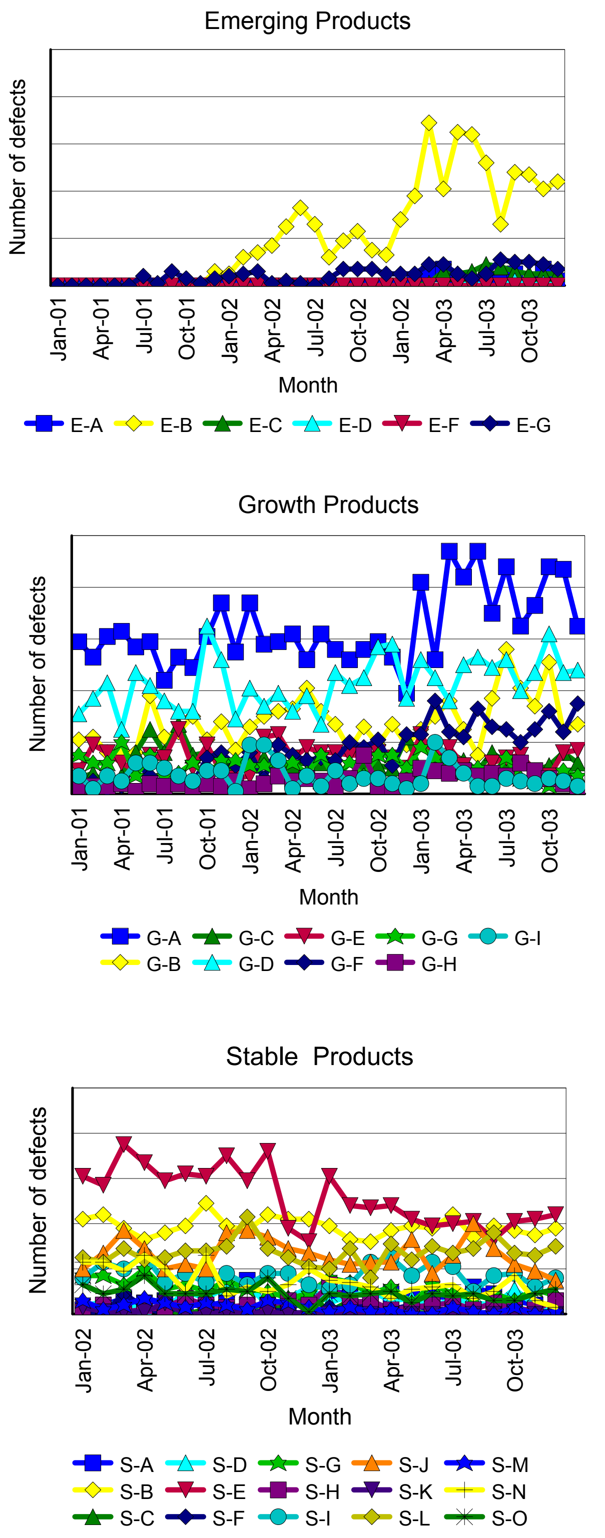

Figure 2: Trend in defects

Note: Even though the ordinate is not labeled, we have kept the same scale for the three graphs above to facilitate easier comparison.

	Problems	Defects
Emerging	54%	32%
Growth	2%	3%
Stable	-1%	1%

Table 2: *Average of month to month change*

necessarily be the focus of an emerging product). For Growth products trend for field problem receipts over time is flattening suggesting that quality is reasonably good. The trend for Stable products is decreasing implying improving quality. This is aligned with our heuristic-based framework described in section 2.

Now, let us look at the volume of problem reports and defects. In Figure 1, even though the ordinate is not labeled for confidentiality purposes, the scale is the same for each of the three charts. Since the install base is small for Emerging products and grows as the maturity evolves, the volume of problem reports and defects is smaller for Emerging products. Hence, it is the trend that determines the quality of the products and not the volume at a given time period. This observation is contrary to the way many organizations think of quality - in terms of volume of problem reports and defects. Our experience shows that the volume alone is not an indicator of quality as volume is dependent on the number of users/licences of the product. In reality, the number of licenses and the extent of actual usage are difficult to track due to the varied marketing channels, accounting practices and inability to verify actual usage. This problem is less acute in managing custom applications where this can be tracked more easily, compared to commercial software program products.

We also analyzed trend by studying the month to month change in field problems and defects for the 3 maturity categories. Table 2 shows that for Emerging products the average month to month change is significantly higher than that for any other type of product, meeting our expectations that Stable products are more reliable. This reflects that the product is improving as it matures. Furthermore, improved documentation and product information and education results in our customers becoming more knowledgeable in their usage of the products over time, and have fewer usage problems.

The above analyses led us to further analyze our data to see if all of our products were categorized correctly. We found that 2 of the 27 products may have been misclassified based on the trend analysis. Further detailed analysis of those two products explained the anomalies to better understand how to manage these two products.

Lesson #2: Its not just the defects but better customer experience which leads to higher quality and maturity.

Now let us look at the volume of defects compared to the volume of problem reports. Although the ordinates are not labeled in figures 1 & 2, the maximum value on the y-axis for field problems in figure 1 is higher than that for defects in figure 2. This indicates that quality should be represented by more than just actual defects, including problems that do not cause a change in the code or documentation. The percentage of problem reports that are not defects reduces as maturity increases indicating more field problems result in a change in code or documentation. But the overall percentage is quite significant, independent of the maturity of the product, and if ignored can result in dissatisfaction amongst customers. This important aspect is often missed by many quality modeling approaches [3, 4]. Another observation is that for all groups, the order of the products is the same for field problems trend and defects trend. The message here is that quality should incorporate the customer experience and should focus on decreasing defects based on usage.

Lesson #3: As maturity increases nature of problems becomes more complex.

Further evaluation of the product maturity suggests that as the maturity of the product improves and new releases are shipped our consumers of the product become more experienced with every new release. Hence for mature products, more complex and critical problems may be reported as the simpler problems have already been caught and fixed in earlier releases. More skilled service staff may be needed to resolve field problems being reported on higher maturity products. This results in increased cost for resolution/problem report. This observation made by the product teams that have used our framework at IBM is worth noting to show the depth of analysis that can be done using our framework.

Lesson #4: Time to market is not always critical for all products.

Emerging products target customers who tend to be early adopters of the product such as the technology enthusiasts and visionaries. Whereas the more mature Stable products are used in environments where stability and reliability are critical. To meet the demands of the early leading adopters of technology, time to market is very critical for Emerging products. If Emerging products are late to market, they may lose to the competition even though they may have better quality and/or functionality compared to competition. On the other hard, having frequent releases for Stable products doesn't necessarily increase market share and may sometimes cause customer dissatisfaction due to frequent upgrades. Hence, time to market issues for releases of more mature products is not as critical as it is for initial releases of Emerging products.

The lessons learned presented in section 3 can be very effectively used to manage a product portfolio in many different situations as described in section 4.

4. Value and usage of framework

In this section, we present different usage scenarios for exploiting the framework we have discussed in managing a product portfolio.

Scenario 1: How should a customer manage the deployment of an IT system?

This scenario is from a software customer perspective. It relates to the management of an IT system which requires different layers of software installed, for example, firmware, operating system, middleware, and applications. Each layer may have one or more products of different maturities. For example, the firmware and OS may be Stable products, the middleware product may be Growth and the application(s) may be Emerging. How does an IT support organization manage such a system? In terms of normal maintenance issues such as the impact of applying bug fixes and product upgrades, the overall system has to be thought of consisting of the different layers of the stack with varying maturity. More mature components can be managed with more aggressive fix/upgrade schedules vs. the less mature components which may introduce too big a change to keep the system stable.

Scenario 2: How should a software owner/vendor manage a group of products?

Companies such as Microsoft, IBM, Oracle, etc. are faced with the challenge of managing many products in the same product group (e.g. Operating systems, Office products, databases, networking software, etc.) of varying maturities with different release cycles. Our framework can help executives compare and contrast products of similar characteristics and maturities to make informed decisions.

Emerging products will have higher development cost compared to the revenue due to new functions being developed and high service cost due to growing trend of problem receipts. Stable products, on the other hand, will have a lower development costs to effectively meet customer needs.

A similar analysis can be used by product portfolio owners such as financial institutions (banks, investment companies, insurance, etc.) that need to evaluate the quality and productivity of the application development organization(s). Research shows that about 70% of IT investment goes into managing and maintaining existing applications [12] making it necessary to understand the portfolio in light of Table 1. If historical data grouped by maturity of the applications is available, then the organization can assess the development and support expenses based on the framework we have presented.

Scenario 3: How should a software owner/vendor manage the development/evolution of a single product?

The maturity framework can also be used to plan a single product evolving in its life cycle and moving from one maturity category to the next. When any product is newly introduced, it is most likely an Emerging product. Using the framework described in this paper, if the evolution of the product from Emerging to Growth can be well understood and planned, the groups can be prepared for the changes in service and development costs, expected requirements growth and the other characteristics described in table 1.

5. Conclusion and Next Steps

In this paper, we have described a heuristics-based product maturity framework that can be effectively used to manage a portfolio of products. We illustrated the basic concepts through lessons learned by using the framework for the IBM Software Group product portfolio. We presented some scenarios to show how this approach can be used to manage a portfolio of products or applications. We hope this work stimulates some new thinking in the industry for managing software portfolios based on the software maturity concepts. At IBM, we are already using this framework for managing some of our product portfolios. We hope to continue our work to validate more of the heuristics presented in section 1 to broaden further deployment.

6. References

[1] Moore, G., “Crossing the Chasm”, Harper Business, 2002.

[2] Jones, C., “The Impact of Poor Quality and Canceled Projects on the Software Labor Shortage”, Technical Report, Software Productivity Research, Inc, 1998.

[3] Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D. and Steece, B., “Software Cost Estimation with COCOMO II”, Prentice-Hall, 2000.

[4] Putnam, L.H. and Myers, W., “Measures for Excellence”, Yourdon Press Computing Series, 1992.

[5] Kan, S., “Metrics and Models in Software Quality Engineering”, Addison-Wesley, 1996.

[6] Mendonca, M, “An Approach to Improving Existing Measurement Frameworks in Software Development Organizations”, Ph.D. Dissertation, University of Maryland, Computer Science Department, 1997.

[7] Hodges, B., Skelly, B., Moncol, M., Smith, B., Coyle, K., Stark, G., Moore, D., Leszkowicz, R., “Software Lifecycle Dynamics”, an IBM internal document, Software Group Quality champions working session, St. Louis, Missouri, 2001.

[8] Chulani, S., Santhanam P., Moore D., Leszkowicz B., Davidson G., “Deriving a Software Quality View from Customer Satisfaction and Service Data’, European Software Conference on Metrics and Measurement, Mar 2001.

[9] Buckley, M. and Chillarege, R., “Discovering relationships between service and customer satisfaction”, Proceedings of the International Conference on Software Maintenance, Opio (Nice), France, Oct 1995.

[10] Shaw, J., “Customer Inspired Quality: Looking Backward through the Telescope”, Jossey Bass Inc., Aug 1996.
[11] Chakrapani, C., “How to Measure Service Quality and Customer Satisfaction”, American Marketing Association, 1997.
[12] Broussard, F., “Streamlining Application Configuration for the Enterprise”, IDC white paper, www.idc.com, Jul 2004.