# A Generic Evolutionary Model for Software Engineering Trends

Latifa Ben Arfa Rabai, Yan Zhi Bai, and Ali Mili
*Institut Superieur de Gestion, Université de Tunis, Bardo 2000 Tunisia*
*New Jersey Institute of Technology, Newark NJ 07102, USA*
*Email contact: latifa.rabai@isg.rnu.tn*

**Keywords:** Bottom Up Approach; Intrinsic factors; extrinsic factors; historical trends; software technology trends; successful trends; quantitative models; Profiles of Success; Cross Influences .

***Abstract***

*Many decision-makers in industry, government and academia routinely make decisions whose outcome depends on the evolution of software technology trends. Even though the stakes of these decisions are usually very high, decision makers routinely depend on expert opinions and qualitative assessments to model the evolution of software technology. In this paper, we report on our ongoing work to build quantitative models of the evolution of software technology trends. In particular, we discuss how we took three trend-dependent evolutionary models and merged them into a single (trend-independent) model.*

## 1. Introduction

Many decision-makers in industry, government and academia routinely make decisions whose outcome depends on the evolution of software technology trends. For example, a corporate manager may take decisions pertaining to the adoption of a particular technology, the adherence to a particular standard, the selection of a particular development environment, etc. A government official may take decisions pertaining to mandating a particular standard, adopting a particular technology, or acquiring a particular product. An academic officeholder may take decisions pertaining to curriculum content or to platform adoption. All these decisions carry important stakes for the organizations at hand and sometimes for the objects of the decisions; yet, they are often made with relatively little hard data, without any duly validated models, relying instead on expert opinions and qualitative assessments.

The work we present in this paper aims to develop quantitative models for the evolution of software technology trends. As we envision them, these models should help us understand what factors drive the evolution of a trend, and to what extent. To this effect, we represent the evolution of a trend by a set of relevant factors, including time dependent and time independent factors, use them to collect factual data about historical trends, then use the data to derive some statistical observations.

The history of software technology is replete with examples where technology trends evolve in unpredictable/ irrational ways, giving us ample motivation to develop quantitative models. For

the sake of argument, we consider the following anomalies, taken from the rich history of programming languages.

- Fortran has had much more success and a much deeper impact than Algol, a language from the same generation (late fifties/ early sixties), which is much more structured, much more orthogonal, and much better designed (Etter 1990).
- Pascal, a language that was designed by a lone Professor as a teaching tool in a first programming course, was much more successful and had a much deeper impact than PL1, a language of the same generation (mid to late sixties), that was developed and promoted by IBM, one of the most influential organizations in the computing field at the time (Niklaus Wirth, 1975).
- The C programming language, a special purpose (systems oriented) language developed with limited ambitions (to accompany a nascent operating system) by a lone researcher has evolved into a major milestone in programming language design, influencing a wide range of subsequent languages, including C++, C#, Object-C, BitC, D, Java, JavaScript, Perl, PHP, etc (Brian 1978).
- Despite being developed as part of a worldwide competition (in the late seventies), despite embodying the most advanced concepts of its time (ADT's, exception handling, genericity, specification vs implementation, etc), and despite enjoying the long term backing of one of the most powerful governmental organizations in the world (the US Department of Defense), Ada had very limited success and made relatively little lasting impact on the discipline of programming language design or the discipline of software engineering (Grady 1987).
- Despite being the focal point of a worldwide research effort in fifth generation computing (eighties and early nineties), despite boasting significant attributes in terms of ease of use, and despite tireless support from many governmental agencies worldwide, Prolog had very limited success as a programming language, and is used in precious few applications (Michael 1994).
- Even though it was designed by a lone researcher, as a language to support a specific project on a very specific computing devise (a set-top device), Java has evolved into a very widely used programming languages, that is widely adopted as a teaching tool, and is a virtual standards for web applications (Jon 2005).

The history of operating systems is no less rich in paradoxes, with systems such as Unix, Linux and DOS starting from relatively modest means to become great successes, whereas systems such as Multics and OS 360 emerging with massive backing from influential organizations in industry, government and academia, to end up with relatively little impact in the long range. A decision-maker in industry, government, academia would be forgiven for betting on the wrong horse when the laws that determine success or failure are so mysterious: success arises in the most unlikely, most modest quarters, and eludes the most likely, best supported contenders (Johnston 2005).

The models we discuss in this paper are primarily empirical, rather than analytical, hence they will not give any insights into how these anomalies came about. What our models try to do, instead, is attempt to capture all the relevant factors that determine the evolution of a software technology trend, and attempt to derive evolutionary laws based on statistical observations.

In section 2 we briefly discuss alternative approaches to modeling software technology evolution and outline the main attributes of the approach we propose. In section 3 we present the empirical background of our project, and in section 4 we present our quantitative approach, along

with its preliminary results. In the conclusion, we summarize and assess our main findings, then outline directions of future research.

## 2. Approaches to Modeling Software Technology Trends

We distinguish, broadly, between two families of approaches to modeling the evolution of software engineering trends; we study them below, in turn.

### 2.1. Top down Approach

The first approach we have considered can be characterized as being analytical, and proceeding top down. This approach breaks down the lifecycle of a product or idea into three partially overlapping phases (Cowan et al. 2002): Research phase, Technology phase, and Market phase. We explore evolutionary models for each phase.

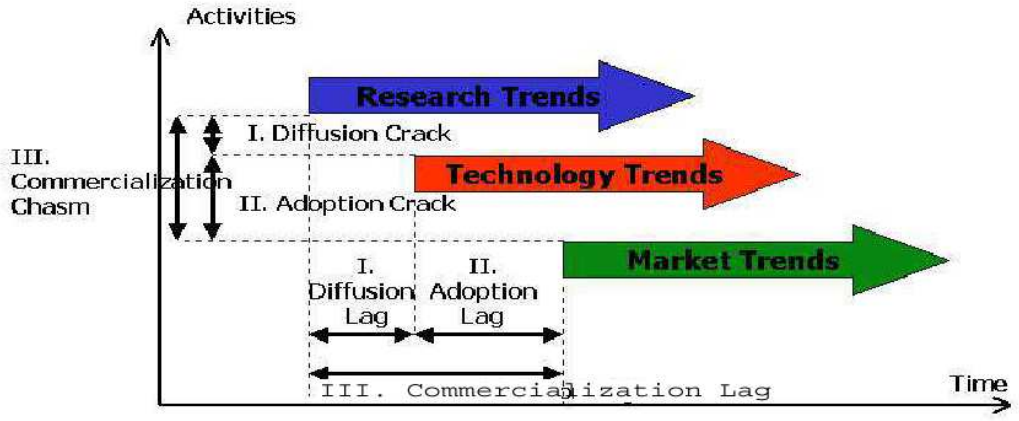


***Figure1: Generic Evolutionary Cycle. (Cowan et al. 2002)***

- Research phase. To model this phase, we have considered research on epistemology (Rogers 1995; Kuhn 1962) and tried to specialize it to Software.
- Technology phase. To analyze this phase, we have considered models of technology evolution and technology transfer (Gaines 1995; Raghavan et al. 1989, Redwine et al. 1985).
- Market phase. To analyze this phase, we have considered models of market trends, such as the *Chasm Model* (Moore 1999), the *Gold Rush Model* (McConnell 1999), and the *Technology Maturation Model* (Redwine et al. 1985).

The X axis represents time, whereas the Y axis represents activities that must take place in order for the trend to proceed through its evolutionary cycle. The various lags are the time periods that various adoption processes take; the various gaps/ chasms are the activities that must take place in order for the trend to proceed successfully. Some trends fail because the corresponding chasms are never crossed. More details on this model can be found in (Cowan et al. 2002).

### 2.2. Bottom Up Approach

To complement the insights gained from the top down approach, we have also considered a bottom up empirical approach, which builds specific evolutionary models from empirical historical data. To this effect we have considered, in turn, three specific families of software artifacts, namely

- ***Programming Languages.***
- ***Operating Systems.***
- ***Middleware Systems,*** which are software layers that serve as an intermediary between systems software and an application. A common application of middleware is to allow programs written for access to a particular database to access other databases (Bai 2009).

To build a quantitative evolutionary model for these families of artifacts, we proceed as follows:

- For each family (programming languages, operating systems, middleware systems), we define a sample of representative elements.
- We define a set of intrinsic factors, which reflect the technical attributes of each member of the family. While the precise definition of these factors depends on the family, they all revolve around the general themes of performance, usability, effectiveness, efficiency, usefulness, generality, cost, etc.
- We define a set of time-dependent extrinsic factors, which reflect the evolving environment in which the members of the family evolved. Whereas intrinsic factors depend on the product family, extrinsic factors are the same for all families of product, and include: *institutional support*, which reflects how much support the software technology/ trend is finding in academic institutions and research laboratories; *industrial support*, which reflects the amount of support the software technology is getting in industry; *governmental support*, which reflects whether or not and to what extent the software technology is supported by governmental agencies; *organizational support* which reflects the support of professional organizations for software technology; and finally *grassroots support*, which reflects the support of professionals and practitioners for the software technology.

The intrinsic factors are used to capture the intrinsic technical attributes of each artifact, and are time-independent; as we discussed earlier, the history of software technology is replete with examples of trends that fail despite having excellent technical attributes, and artifacts that fail despite being very mediocre. Hence there is more to success and failure than technical merit. The extrinsic factors, which, by contrast with intrinsic factors, are dependent on time, are used for two purposes:

- First to model the environment in which a trend evolves (how much support it gets from the various quarters of the community of stakeholders);
- Second to model multi-dimensional metrics of success; we do not say, this trend is successful or is not successful. Rather, we present the vector of extrinsic factors of the trend at a given date, and we let that vector speak for itself (anyone who sees the vector of extrinsic factors can decide for herself/ himself whether it represents a successful trend or not, depending on the specific prioritization of the extrinsic factors and the specific standards applied on each extrinsic factor).

Using this quantitative information, we build statistical models that take the intrinsic factors and past extrinsic factors as independent variables and the present or future extrinsic factors as

dependent variables. These models allow us to predict the evolution of a trend on the basis of its intrinsic attributes and the historical evolution of its extrinsic attributes.

## 3. Research Background

### 3.1 Programming Languages

To analyze the evolution of programming languages, we have considered a sample of 17 programming languages, including: ADA, ALGOL, APL, BASIC, C, C++, COBOL, EIFFEL, FORTRAN, JAVA, LISP, ML, MODULA, PASCAL, PROLOG, SCHEME, and SMALLTALK. The intrinsic factors we have defined for programming languages include: Reliability, Extensibility, Expressiveness, Generality, Orthogonality, Machine independence, Efficiency, Simplicity, Maintainability and Implementability. This work was completed in 2003 and collected quantitative information on the extrinsic factors for 1993, 1998, and 2003. A sample of the results we obtain from our statistical analysis is given in Figure 2 (Chen 2005). The values for 2008 were derived using the predictive model.

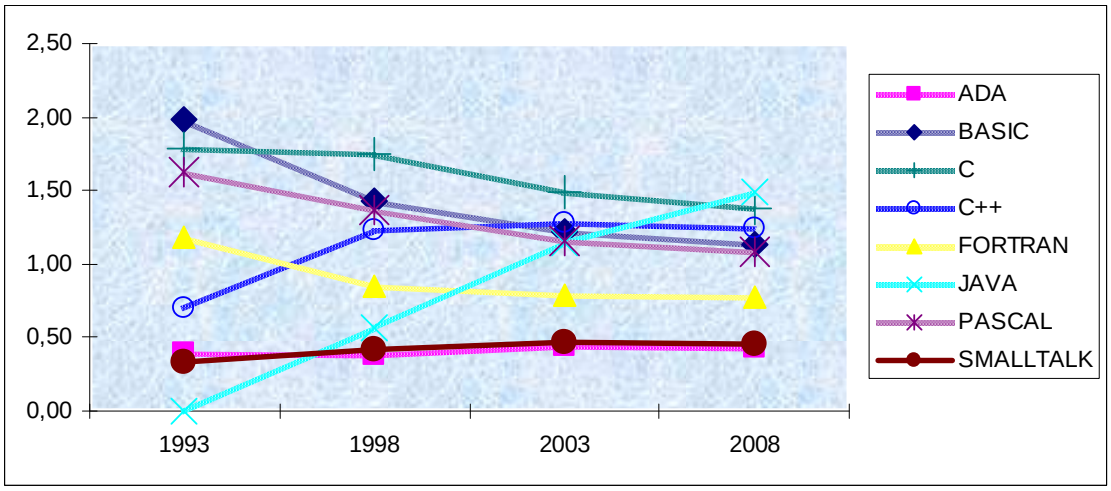


***Figure 2: Evolution of Grassroots support for programming languages: Actual (1993-2003) and predicted (2008)***

### 3.2. Operating Systems

To analyze the evolution of operating systems, we have considered a sample of 15 operating systems, including: UNIX, Solaris/Sun OS, BSDs, OS/2, Windows, MS-DOS, MAC OS, Linux, NetWare, HP-UX, GNU Hurd, IBM AIX, Compaq/DEC, VMS, Multics, and OS360. The intrinsic factors we have defined for operating systems include: Security and Protection, Reliability, Portability, Compatibility, Openness, Design, Scalability, Ease of learning, Ease of use, Consistency of Interaction Protocols, Cost, CPU Management, Memory Management and IO Management. This work was completed in 2004 and collected quantitative information on the extrinsic factors for 1997, 2000, and 2003. A sample of the results we obtain from our statistical analysis is given in Figure 3 (Peng 2007). The values for 2006 were derived using the predictive model.

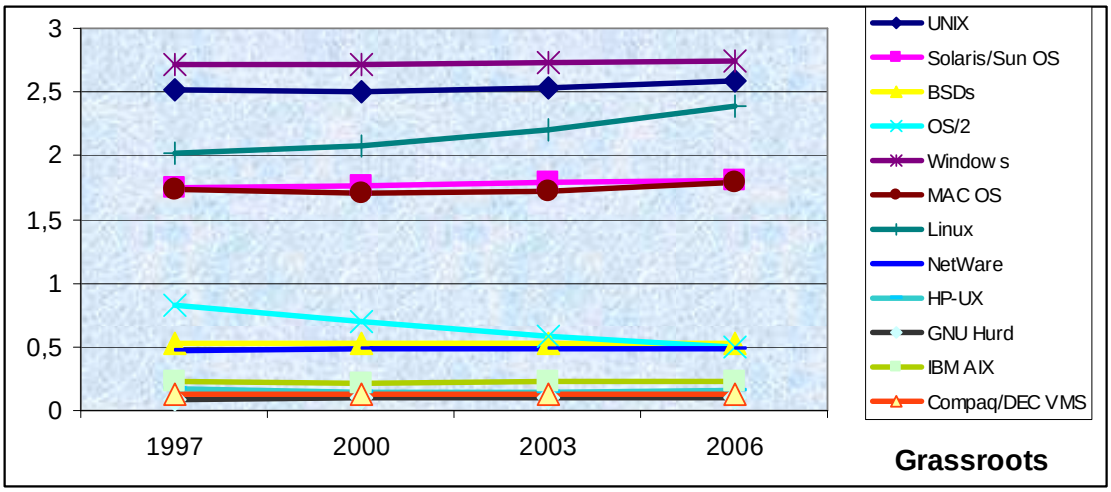


***Figure 3: Evolution of Grassroots support for operating systems: Actuals (1997-2003) and Predicted (2006)***

### 3.3 Middleware Systems

To analyze the evolution of middleware systems, we have considered a sample of 18 middleware systems, including:

- ODBC and JDBC is a middleware that connects programming languages (Java, as well as Windows based programming languages) to databases.
- JavaBean is a middleware that allows programmers build reusable application blocks called components that can be deployed in a network on any major operating system platform.
- EJB (Enterprise Java Bean), is a managed, server side component for modular construction of enterprise applications.
- COM/COM+/DCOM, is a middleware supported by Microsoft, that provides heterogeneity across languages on the Windows operating system.
- CORBA (Common Object Request Broker Architecture), is a middleware produced and supported by the Object Management Group, that allows remote method invocations on objects; it offers heterogeneity across programming language and vendor implementations.
- Jini is an architecture for distributed computing, that addresses how to provide object services to clients, and how these clients find available services. A Jini service may control a piece of hardware, using the Java Native Interface (JNI).
- JMS (Java Message Service) , is a middleware that provides integration services with existing messaging systems.
- MSMQ is essentially a messaging protocol from Microsoft that allows applications running on disparate servers to communicate in a failsafe manner.
- MQSeries is an IBM software family whose components are used to tie together other software applications so that they can interoperate.

- MTS, the Microsoft Transaction Server is a program that runs on a network server and manages application and database transaction requests on behalf of a client computer user.
- Dot NET, The Microsoft NET Framework is a software component that can be added to the Microsoft Windows operating system. It provides a large body of pre-coded solutions to common program requirements, and manages the execution of programs written specifically for the framework.
- J2EE is an industrial standard /product initiated by Sun Microsystems. It specifies a programming platform for developing and running distributed multi-tiered architecture Java applications, based largely on modular software components running on a special server.
- JBoss Enterprise Middleware Suite (JEMS) is an extensible and scalable suite of products for creating and deploying e-business applications.
- The BEA WebLogic Platform which includes BEA WebLogic Server®, BEA WebLogic Portal™, BEA WebLogic Integration™, BEA WebLogic Workshop™, BEA JRockit™, is an application platform suite that helps developers to service-enable their applications.
- IBM WebSphere as a brand refers to a group of IBM software products. From a technical perspective, WebSphere typically means the WebSphere Application Server or WAS product . WAS provides a set of services , such as database access, mail services and security services.
- Apache Geronimo is a middleware whose goal's is to produce a server runtime framework that pulls together the best Open Source alternatives to create runtimes that meet the needs of developers and system administrators and a family of products ranging from application development tools and integration solutions to identity management, collaboration, and business intelligence reporting .

The intrinsic factors we have defined for middleware systems include: Availability, Security and Protection, Maintenance and management, Performance, Interoperability, Scalability, Support for existing applications, OS supported, Languages Supported, Standard Support, Ease of learning, Ease of use, Operation Cost, Acquisition Cost, Tools supporting development and management and Breadth of applicability. A sample of the results we obtain from our statistical analysis is given in Figure 4 (Bai 2009).

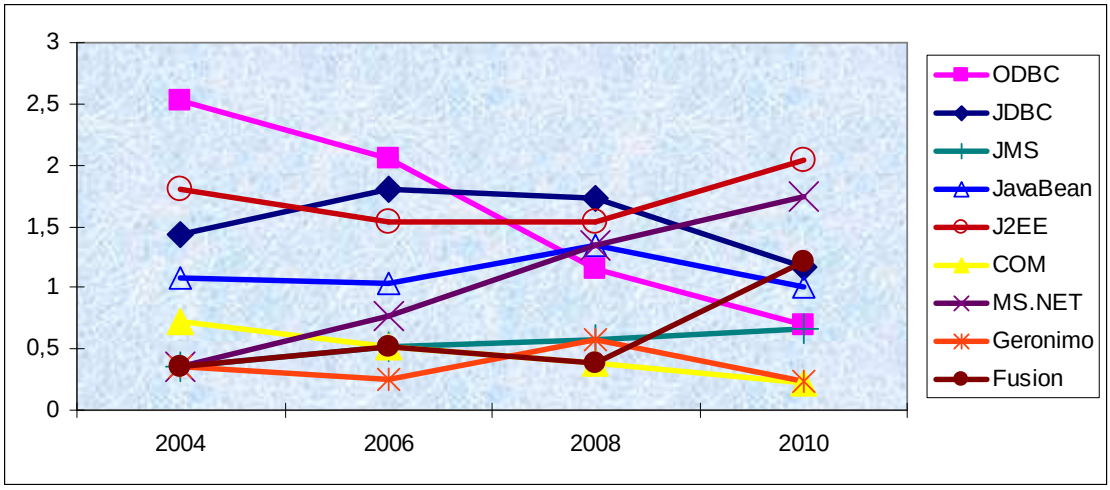


***Figure 4: Evolution of institutional support from 2006 to 2010: Actual Values (2004-2008) and Predicted Values (2010)***

## 4. A Generic Evolutionary Model

Each of the models we have derived so far can be used to understand the evolution of a specific family of products (programming languages, operating systems, middleware systems). For example, to chart the future evolution of a programming language, we feed the values of its intrinsic factors (simplicity, genericity, orthogonality, machine-independence, expressiveness, etc) along with the history of its extrinsic factors (institutional support, governmental support, industrial support, etc) into our model, and we derive future predictions for the extrinsic factors (how much future support the language will enjoy from each set of stakeholders). To chart the future evolution of an operating system, we apply the predictive model of operating systems, using OS-specific intrinsic factors, such security, portability, scalability, resource management, etc. To chart the future evolution of a middleware system, we apply the predictive model of middleware systems, using middleware-specific intrinsic factors, such performance, interoperability, OS supported, languages supported, ease of learning, ease of use, tool support, breadth of applicability, etc. How about if we are interested in charting the future evolution of, web browsers? Or database management systems? Or programming environments? Or mail clients? We cannot keep developing evolutionary models for each conceivable family of products.

What we have resolved to do instead is to use the data we have collected for these three distinct models to build a generic model for a wider range of software products. To do so, we need to define trend-independent intrinsic factors, and map the trend-specific intrinsic factors of the three past studies onto these; then we can merge all our data sets and analyze them anew. This approach, which we detail in this section, offers us two advantages, but also carries some risk:

- The advantages include: First, that we obtain a model that can, in principle, be applied to a broader range of software trends, since its intrinsic factors are generic. Second, that we now have a broader sample of data to use for our statistical analysis (50 elements in our statistical sample).

- The drawback of this resolution is that because they are generic, the intrinsic factors carry less information about the software products we are applying them to.

Under the circumstances, this is a tradeoff we are prepared to make.

### 4.1. A Research Plan

In this section we discuss our plan to combine the three specific evolutionary models to derive a generic model that can be applied to any software technology. To this effect, we proceed as follows:

1. We define a set of generic intrinsic factors that are trend independent, i.e. applicable to any software product/ technology. Whereas attributes such as genericity, strong typing, and expressiveness apply only to programming languages; whereas attributes such as CPU management, I/O management, Deadlock management apply only to operating systems; and whereas attributes such as interoperability, scalability, and range of supported languages apply only to middleware systems; the attributes we choose for the generic model apply to all software technologies/ products. These include: operational usefulness, functional usefulness, usability, versatility, and cost. Of course, these are not as meaningful as the trend-specific factors, but for the sake of broad applicability we trade significance for generality. Also, for the sake of being able to apply statistical analysis, we ensure that all these factors are quantifiable.
2. We map all trend specific factors onto trend-independent factors for any software technology; these have been defined in such a way as to encompass all trend specific factors.
3. From the mapping, of trend-specific factors to trend-independent factors, we infer normalized values for the generic intrinsic factors of all the products we have studied, whether they are programming languages, operating systems, or middleware systems.
4. For extrinsic factors, we determine the periodicity of historical data, and we record the values of all historical data on a common periodicity, by appropriate interpolations and extrapolations. We have chosen the periodicity to be two years; hence for each product we record extrinsic factor values for the present, two years ago, four years ago and six years ago.
5. We build a data table with all the individual products, along with numeric values for all their intrinsic and extrinsic factors (which are identical for all studies, including the generic study).
6. We derive quantitative statistical models that relate the current or future values of extrinsic factors as a function of the intrinsic factors and the history of extrinsic factors.
7. To validate the generic predictive model that we obtain, we are currently conducting an independent empirical study on two technologies, data bases and web browsers, using the generic intrinsic factors, the common extrinsic factors, and the periodicity determined in step 4; and we use the results of this data to validate the model derived in step 6.

At the time of this writing, step 7 is under way, steps 1 through 6 are completed. The data table alluded to in step 6 is available online at http://techwatch.urpah.net/tecgeneric.xls.

*Table1: Mapping Trend Specific Intrinsic Factors onto Generic Factors*

| | MW. Intrinsic factors | | OS. Intrinsic factors | PL. Intrinsic factors |
|---|---|---|---|---|
| **Attribute Names** | **Attribute Names** | **Sub-attributes** | **Sub-attributes** | **Sub-attributes** |
| **Usefulness Operational** | Operational Quality | Availability<br>Security and Protection<br>Maintenance and management<br>Performance | Reliability<br>Security & Protection | Reliability<br>Extensibility<br>Expressiveness |
| **Versatility** | Generality | OS supported<br>Languages Supported:<br>Standard Support:<br>Support for existing applications<br>Interoperability<br>Scalability | Portability<br>Compatibility<br>Openness<br>Design<br>Scalability | Generality<br>Orthogonality<br>Machine independence |
| **Usability** | Usability | Ease of learning<br>Ease of use | Ease of learning<br>Ease of use<br>Consistency of Interaction Protocols | Efficiency<br>Simplicity |
| **Cost** | Cost | Acquisition Cost<br><br>Operation Cost | Cost<br><br>CPU Management<br>Memory Management<br>IO Management | Maintainability |
| **Usefulness functional** | Functionality | Breadth of applicability<br>Tools supporting development and management | Range of Services<br>Languages Support<br>Distributed Computing<br>Network Services<br>Deadlock Management | Implementability |

## 4.2. Regression Model for Historical Trends

We model the degree of success of a trend, not by a number, but by a vector of extrinsic factors that reflect its popularity with different quarters of the technology scene: government agencies, industrial organizations, academic institutions, professional bodies, and end users (grassroots). We anticipate that these factors influence each other over time: a trend that is widely followed in academia one year may spread to industry several years later when students graduate and work in industry; conversely, a trend that is widely followed in industry one year may find its way in academic programs years later due to pressure from recruiters or from shifting job markets; a trend that is widely popular with grassroots practitioners one year may gain industrial support later through market pressure; etc. To model all these complex interactions, we consider the intrinsic factors of each trend, along with the history of its extrinsic factors, and we build a regression model that derives the values of the extrinsic factors of a trend at year *Y* as a function of the intrinsic factors of the trend as well as the history of the extrinsic factors of that trend in past years. We build a linear regression equation for each extrinsic factor; the dependent variable of each regression is the relevant extrinsic factor, and the independent variables are the (time-independent) intrinsic factors, as well as the history of past extrinsic factors, of the trend.

We build this model by feeding it past and present extrinsic data, and we use it as a predictive tool by feeding it past and present data and querying it on future data. Specifically, if we let $E_y$ be the vector of extrinsic factors of a trend at year y, *I* be the vector of (time-independent) intrinsic factors of the trend, then the regression function provides us with the optimal values of $\alpha$, $\beta$, $\gamma$, and $\delta$ that minimize the error term $\varepsilon$ in the equation:

$$E_{2009} = \alpha * I + \beta * E_{2007} + \gamma * E_{2005} + \delta E_{2003} + \varepsilon .$$

Using this regression model, we can predict the future of extrinsic factors by applying the model to past and present data, as shown below:

$$E_{2011} = \alpha * I + \beta * E_{2009} + \gamma * E_{2007} + \delta E_{2005} + \varepsilon$$

where $\alpha$ the parameter matrix for intrinsic factors, $\beta$ the parameter matrix for extrinsic factors in 2009, $\gamma$ the parameter matrix for extrinsic factors in 2007, $\delta$ the parameter matrix for extrinsic factors in 2005 and $\varepsilon$ an error term.

## 5. Profiles of Success

When we use our predictive model to plot the future evolution of a set of trends with respect to a particular extrinsic attribute, our result is contingent on the future evolution being a continuation of past evolution. But this hypothesis is not always borne out in practice, where accidents (in the form of unanticipated trends, novel ideas, disruptive events) can create singularities in the evolutionary process. To make our predictive model immune to such discontinuities, we shy away from making predictions about individual artifacts, and make predictions instead on the profile of successful artifacts. For example, we do not make statements such as: *"Java will be adopted by 40% of*

*grassroots in 2011*"; rather we will make statements such as: "*The language that will be most popular with grassroots users in 2011 will have the following attributes, ranked from most important to least important*". The latter statement is much safer than the former, as it characterizes successful languages by their intrinsic attributes rather than to identify them by name, and may include a language we may not even know today.

How do we do this? We proceed in three steps:

- ***Quantify correlations between intrinsic factors and past extrinsic factors***. For each past year and each extrinsic factor, we compute and plot the correlation between intrinsic factors and the selected extrinsic factor at the selected year. This correlation can be interpreted as the extent to which the intrinsic factor was important for success with respect to the extrinsic factor at the selected year; for example (see Figure 5), in 2007, the correlation between functional usefulness and grassroots support was 0.45.
- ***Consider the past evolution of these correlations.*** This past evolution enables us to determine, for each extrinsic factor, how intrinsic factors are correlated to success or failure, and how this correlation is evolving over time. For example (see Figure 8), operational usefulness is increasingly important to success with respect to governmental support.
- ***Project this Evolution into the future.*** Use regression techniques to generate regression formulas that can be applied to predict future values of the correlation between intrinsic values and extrinsic values in future years. This allows us to estimate how much importance each intrinsic factor has with respect to each extrinsic factor in future years. For example, we can infer from Figure 6 that to be successful with respect to institutional support (in academia), a trend needs to have the following attributes, by order of decreasing importance: cost, operational usefulness, versatility, functional usefulness, and usability.

This creates what we call *Profiles of Success.* We have one such a profile for each extrinsic factor, shown in the Figures below (Figures 5 to 9). Because ours is primarily an empirical effort rather than an analytical effort, we do not provide an explanation for the observation, but content ourselves with making observations, upon checking that they are statistically well-founded. Some of the observation sound counter-intuitive, and indeed are hard to explain (for example, negative correlations between intrinsic factors and extrinsic factors); but we argue that the phenomena we describe are themselves replete with paradoxes, as we discussed in the introduction. Be that as it may, we limit ourselves, in interpreting these curves, to general statements about broad trends. For example, we use the curves to make statements such as:

- In 2011, we expect intrinsic factors IF and IF' to be the two most important factors of success with respect to extrinsic factor EF.
- In 2011, we expect intrinsic factors IF and IF' to be the two least important factors of success with respect to extrinsic factor EF.
- In 2011, we expect intrinsic factor IF to be more important than intrinsic factor IF' in ensuring success with respect to extrinsic factor EF.
- The importance of intrinsic factor IF with respect to the success criterion defined by extrinsic EF increases from 2003 to 2011.

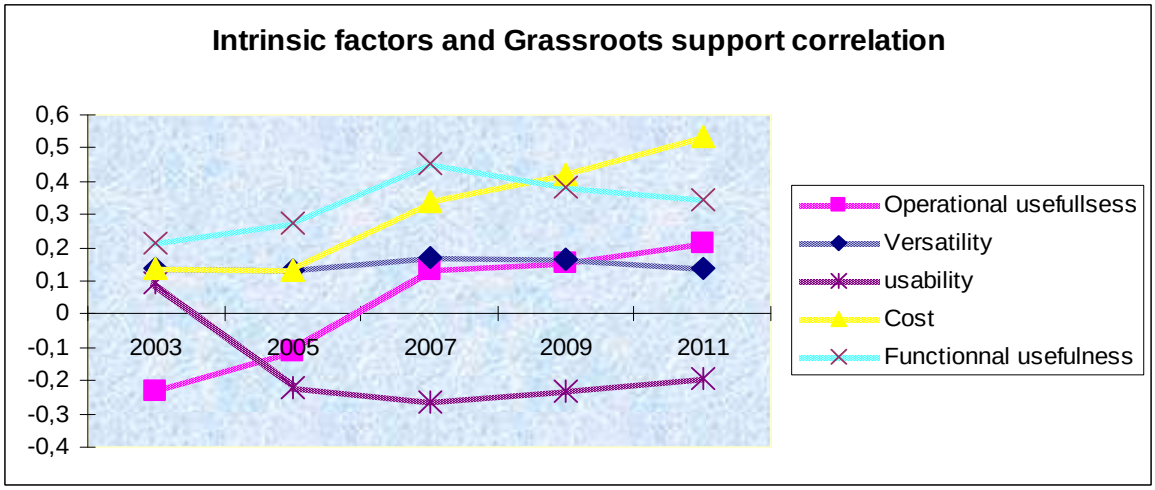


***Figure 5: Profiles of Success with respect to Grassroots support***

To be successful with grassroots (independent users) in 2011, a software product has to have low cost, high functionality, and high operational attributes.

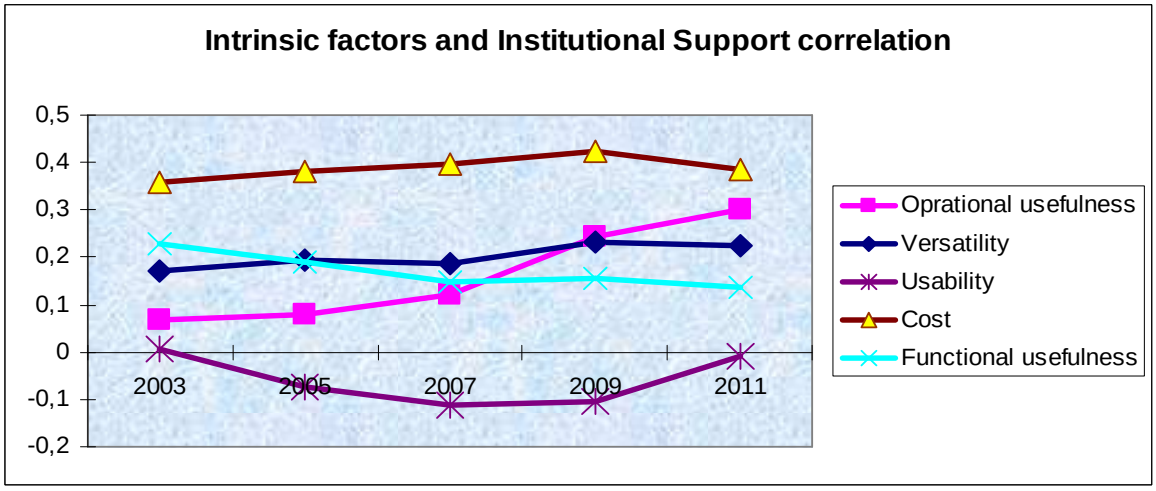


***Figure 6: Profiles of Success with respect to Institutional support***

To be successful in academia in 2011, a software product has to have low cost, high operational attributes, and high versatility.

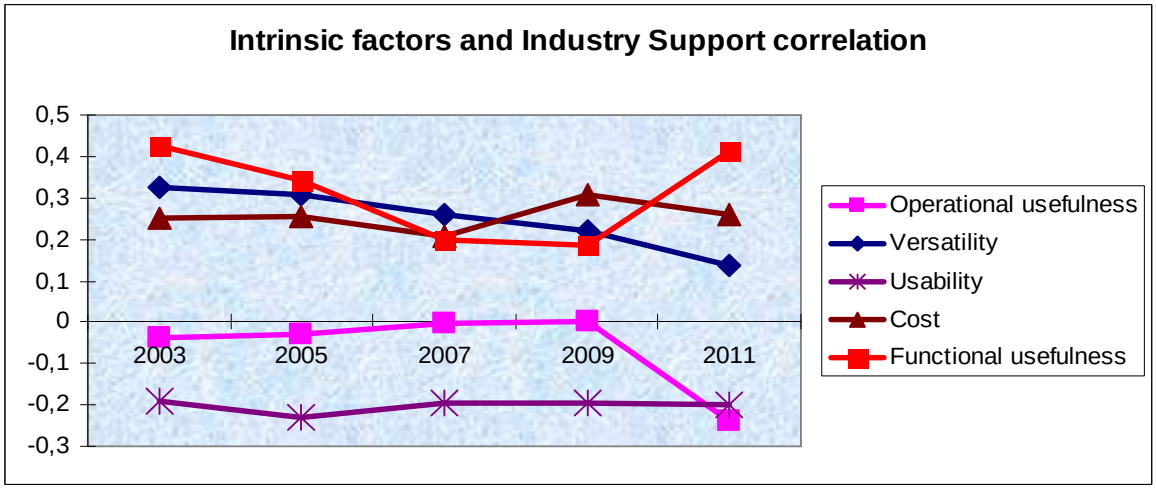


*Figure 7: Profiles of Success with respect to Industrial support*

To be successful in industry in 2011, a software product has to have superior functionality, low cost, and high versatility.

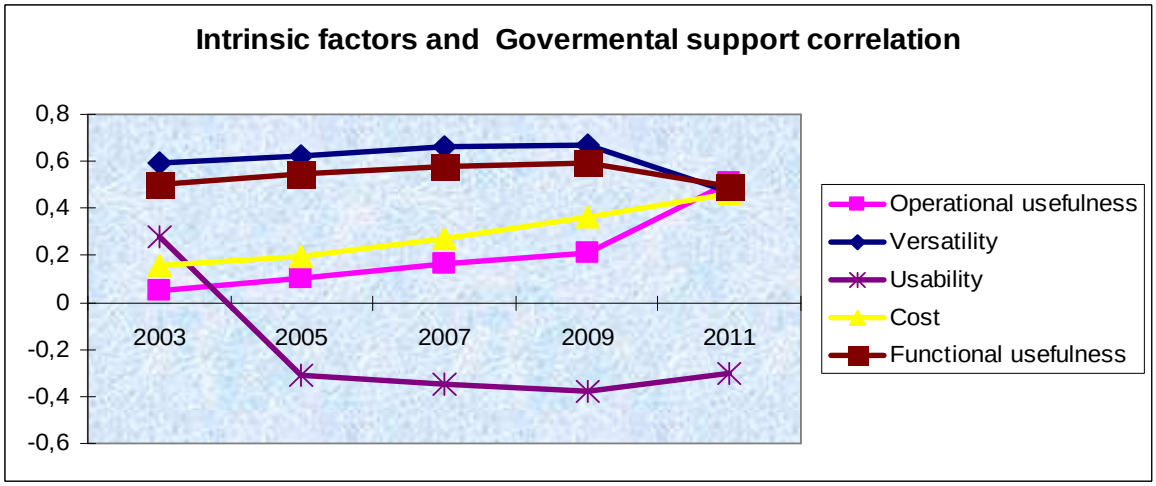


*Figure 8: Profiles of Success with respect to Governmental support*

To be successful in governmental agencies in 2011, a software product/ trend will have to have high operational attributes, high versatility, and high functionality.

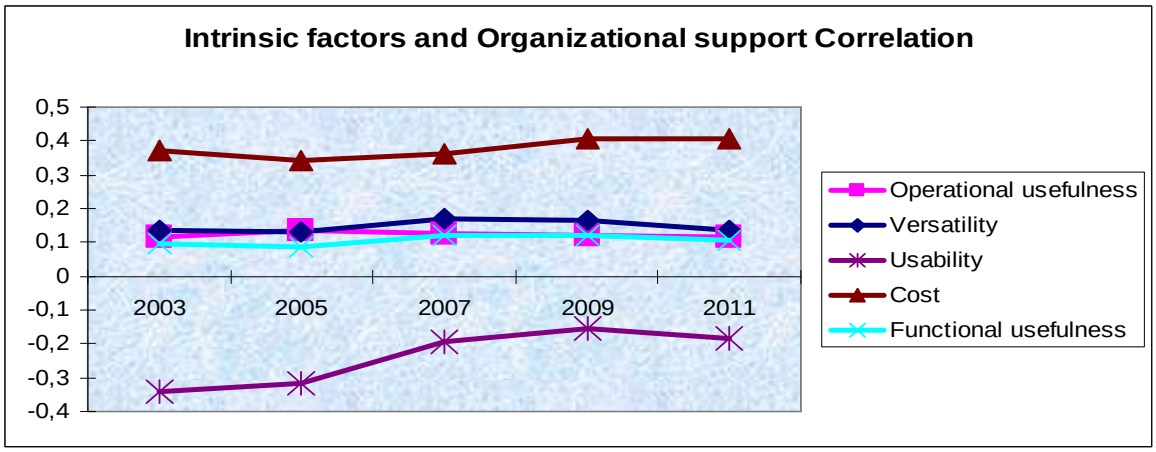


***Figure 9: Profiles of Success with respect to Organizational support***

To be successful with professional bodies in 2011, a software product/ trend will have to have low cost, high versatility, and high operational attributes (availability, reliability, extensibility, etc).

To complement the insights provided by the study of correlations between intrinsic factors and extrinsic factors, we have also sought to compute statistical regressions using the extrinsic factors as dependent variables and the intrinsic factors as independent variables. We use regression coefficients to assess to what extent a given intrinsic factor contributes to the success of a trend with respect to an extrinsic factor. By studying these regressions for several time periods, we can plot the evolution of the impact that an intrinsic factor has on an extrinsic factor. Not all the regressions produced statistically significant results, and those that did do not produce results that are significantly distinct from what we know from correlations.

## 6. Modeling Cross Influences

The evolution of software technology trends involves several stakeholders, who influence each other in a variety of ways, such as:

- An artifact that is successful in academia may find its way to industry when students trained with the artifact graduate and move to industry (e.g. Pascal).
- An artifact that is successful in industry may find its way to academia through pressure from recruiters and conditions of the job market (e.g. C, UML).
- An artifact that is successful with grassroots may find its way to industry when industrial organizations see a potential market (e.g. Linux).
- An artifact that is successful with governmental organizations may find its way to academia through funding incentives and funding restrictions (e.g. Ada).

In order to quantify these cross-influences, we analyze the correlations between histories of extrinsic factors and present values of other extrinsic factors. We review and comment on some of these cross influences.

### 6.1 From Academia to Industry

We consider correlations between past institutional support and current industry support, and find the following correlation matrix:

| | Industrial | Institutional, -2 | Institutional, -4 | Institutional, -6 |
|---|---|---|---|---|
| **Industrial** | 1.0000 | | | |
| **Institutional, -2** | 0.8440 | 1.0000 | | |
| **Institutional, -4** | 0.7780 | 0.9750 | 1.0000 | |
| **Institutional, -6** | 0.6678 | 0.8925 | 0.9666 | 1.0000 |

***Table 2: Modeling Cross Influences From Academia to Industry***

The correlation values in this matrix reflect the extent to which past success in academia (institutional -2, two years ago; institutional-4, four years ago, institutional-6, six years ago) affects current success in industry. The monotonicity of the first column of the table suggests that the impact of academic success over industrial success fades with time.

### 6.2 From Industry to Academia

For the sake of comparison with the previous section, we also consider correlations between past industrial support and current institutional support; we find the following correlation matrix:

| | Institutional | Industrial, -2 | Industrial, -4 | Industrial, -6 |
|---|---|---|---|---|
| **Institutional** | 1.0000 | | | |
| **Industrial, -2** | 0.6532 | 1.0000 | | |
| **Industrial, -4** | 0.6743 | 0.8851 | 1.0000 | |
| **Industrial, -6** | 0.5785 | 0.7250 | 0.9584 | 1.0000 |

***Table 3: Modeling Cross Influences From Industry to Academia***

The correlation values in this matrix reflect the extent to which past success in academia (institutional -2, two years ago; institutional-4, four years ago, institutional-6, six years ago) affects current success in industry. By contrast with the first column of Table 2, the first column of this table is not monotonic; except for the first entry, the column reaches its maximum for industrial-4, which seems to suggest that the length of time that a successful trend in industry takes to make its way to academia is around four years..

This table also suggests, in combination with the previous table, that academia influences industry more than the other way around, given that the first column of table 2 has higher values than the first column of table 3.

### 6.3 From Industry to Grassroots

We consider correlations between past industrial support and current grassroots support, and we find the following correlation table.

| | Grassroots | Industrial, -2 | Industrial, -4 | Industrial, -6 |
|---|---|---|---|---|
| **Grassroots** | 1.0000 | | | |
| **Industrial, -2** | 0.3757 | 1.0000 | | |
| **Industrial, -4** | 0.5026 | 0.8851 | 1.0000 | |

| | | | | |
|---|---|---|---|---|
| **Industrial, -6** | 0.5006 | 0.7250 | 0.9584 | 1.0000 |

***Table 4: Modeling Cross Influences From Industry to Grassroots***

There are no explicit mechanisms by which industry directly influences grassroots choices; this may explain why the correlation we find in this matrix are significantly lower than those we find between industry and academia, or between academia and industry.

### 6.4 From Government to Academia

We consider correlations between past governmental support and current institutional support, and we find the following correlation table.

| | **Institutional** | **Governmental, -2** | **Governmental, -4** | **Governmental, -6** |
|---|---|---|---|---|
| **Institutional** | 1.0000 | | | |
| **Governmental, -2** | 0.7175 | 1.0000 | | |
| **Governmental, -4** | 0.6715 | 0.9941 | 1.0000 | |
| **Governmental, -6** | 0.6343 | 0.9837 | 0.9971 | 1.0000 |

***Table 5: Modeling Cross Influences From Government to Academia***

This table shows some impact of government support on industrial support, perhaps due to incentives and disincentives used by governmental agencies to promote or discourage from technologies; the first column of this table shows that this impact wanes with time.

## 7. Conclusion

The evolution of software technology trends is notoriously difficult to model, leading stakeholders and decision-makers to rely primarily on intuition and expert judgment. In this paper, we attempt to complement these approaches with some insights gained from collecting factual data about actual trends over a period of time, and from analyzing this data to derive empirical evolutionary models.

We have used this model to provide answers to two broad questions:

- *First, what characterizes successful trends of the future?* To answer this question, we plot the evolution over time of the correlation between intrinsic factors and extrinsic factors, and perform a regression that allows us to predict what this correlation will be in the future. Knowing the correlation between a given extrinsic factor and all the intrinsic factors allows to draw the profile of a successful trend, where success is measured by the extrinsic factor in question.
- *Second, how do various stakeholders of software technology influence each other's choices?* To answer this question, we compute correlations between various extrinsic factors and evolutionary histories of other extrinsic factors. The analysis of these correlations informs us on the influence that the success of trend with one quarter has on its success with another quarter.

These preliminary results provide a basis for a quantitative analysis of the evolution of software technology trends.

## REFERENCES


Bai, Y.Z., 2009. *Modeling the Evolution of Middleware Systems.* Tech Report. New Jersey Institute of Technology. 2009.

Brian Kernighan, Dennis Ritchie: *The C Programming Language*. 1st, Prentice Hall 1978; ISBN 0-13-110163-3. Pre-ANSI C.

Etter, D. M.., 1990. *Structured FORTRAN 77 for Engineers and Scientists* (3rd ed.). The Benjamin/Cummings Publishing Company, Inc. ISBN 0-8053-0051-1.

Chen, Y. F., Dios, R., Mili, A., Wu, L. and Wang, K.F. 2005. Programming Language Trends: An Empirical Study. *IEEE Software.* 2005.

Cowan Robert D., Mili, A., Hany, H., Ammar, A., McKendall Jr., Yang, L., Chen, D. and Spencer, T. 2002. Software Engineering Technology Watch. *IEEE Software* 19(4): 123-129 (2002).

Gaines, B. 1995. *Modeling and Forecasting the Information Sciences.* University of Calgary, Alberta.

Grady Booch. 1987. SOFTWARE ENGINEERING WITH Ada, *California: The Benjamin/Cummings Publishing Company, Inc., 1987. ISBN 0-8053-0604-8*

Johnston (April 1, 2005). "VSE: A Look at the Past 40 Years". *Z/Journal* (Thomas Communications, Inc.) (April/May 2005).

Jon Byous, 2005. *Java technology: The early years*. Sun Developer Network, no date [ca. 1998]. Retrieved April 22, 2005.

Kernighan, Dennis M. Ritchie. 1988. *The C Programming Language* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall .

Kuhn, Th. S. 1962. *Structure of Scientific Revolution.* University of Chicago Press.

Michael A. Covington. 1994. *Natural Language Processing for Prolog Programmers*, ISBN 0-13-629213-5

Moore Geoffrey A. 1999. *Crossing the Chasm.* Harper Business.

McConnell, S. 1999. *After the Gold Rush.* Microsoft Press.

Niklaus Wirth: *Algorithms + Data Structures = Programs*. Prentice-Hall, 1975, ISBN 0-13-022418-9.

Peng, Y., Li, F. and Mili, A. 2007. Modeling the evolution of operating systems: An empirical study. *Journal of Systems and Software.* 80(1): 1-15.

Raghavan, S. and Chand, D. 1989. Diffusing Software Engineering Methods. *IEEE Software*, pp 81-90, July.

Redwine Jr., Samuel, T. and Riddle William E. 1985. Software Technology Maturation. *ICSE_*. pp189-200.

Rogers, E.M. 1995. *The Diffusion of Innovations.* 4th ed. The Free Press :New York, NY.