MODELING THE EVOLUTION OF SOFTWARE ENGINEERING TRENDS
A Bottom Up Approach

Latifa Ben Arfa Rabai, Yan Zhi Bai
Institut Superieur de Gestion de Tunis, Université de Tunis, Bardo 2000, Tunisia
New Jersey Institute of Technology, Newark NJ 07102, U.S.A.

Ali Mili
New Jersey Institute of Technology, Newark NJ 07102, U.S.A.

Keywords: Bottom Up Approach, Intrinsic factors, Extrinsic factors, Historical trends, Software technology trends, Successful trends.

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, 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. 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. Because this is ongoing research, we do not present an objective validation of our proposed model, but outline a validation plan nevertheless. 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

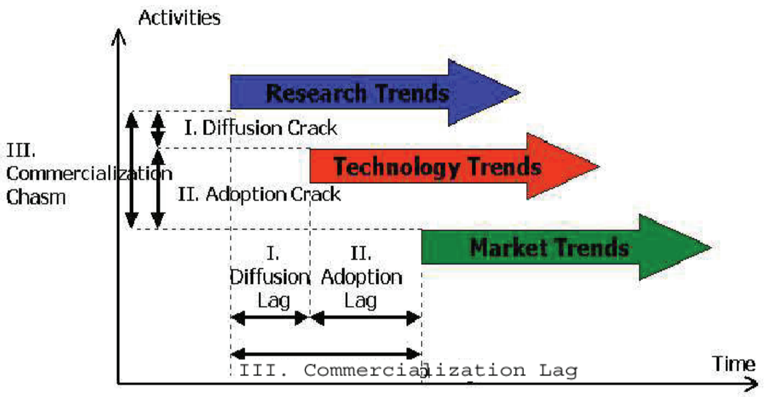

Figure 1: Generic Evolutionary Cycle. (Cowan et al. 2002).

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.

- Research phase. To model this phase, we have considered research on epistemology (Rogers 1995; Kuhn 1996) 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 three specific families of software artifacts, namely Programming Languages.

- Operating Systems.
- Middleware Systems.

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.
- 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 a governmental agency, organizational support which reflects the support of professional organizations for software technology, and grassroots support which reflects the support of professionals and practitioners for the software technology.

Using this quantitative information, we build

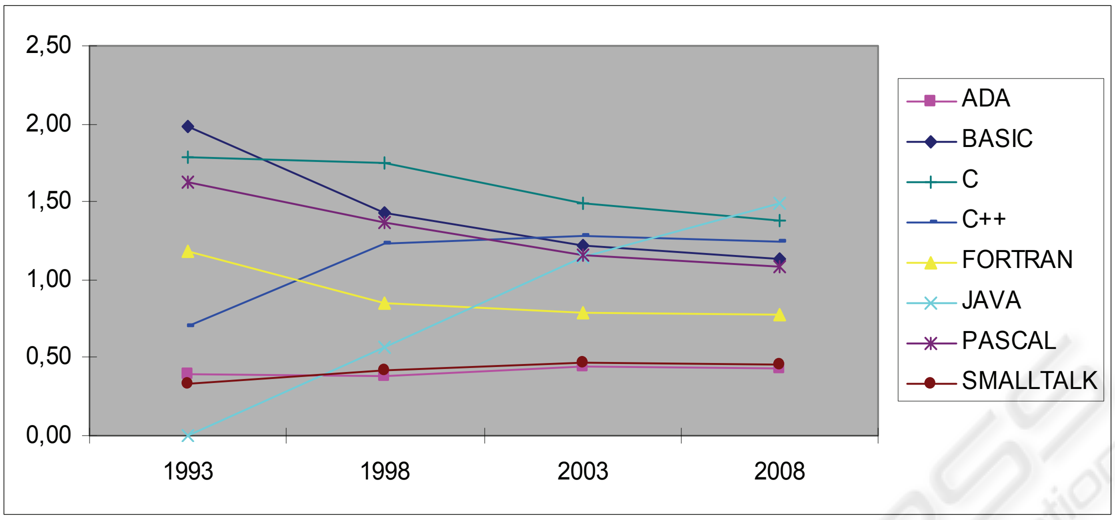

Figure 2: Evolution of Grassroots support for programming languages (Actual and predicted).

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 Implementability. For the sake of simplicity, we assume these factors to be time-independent; of course, it is not uncommon for a language to evolve with time, but we consider that any significant evolution in its intrinsic factor creates a new product rather than an evolution of the existing product. 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 et al. 2005). The values for 2008 were derived using the predictive model. Composed in 2003, this figure shows how the popularity of the various programming languages with grassroots users evolved (as a matter of fact) between 1993 and 2003, and how it was expected to evolve subsequently, up to 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 & 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 et al. 2007). The values for 2006 were derived using the predictive model.

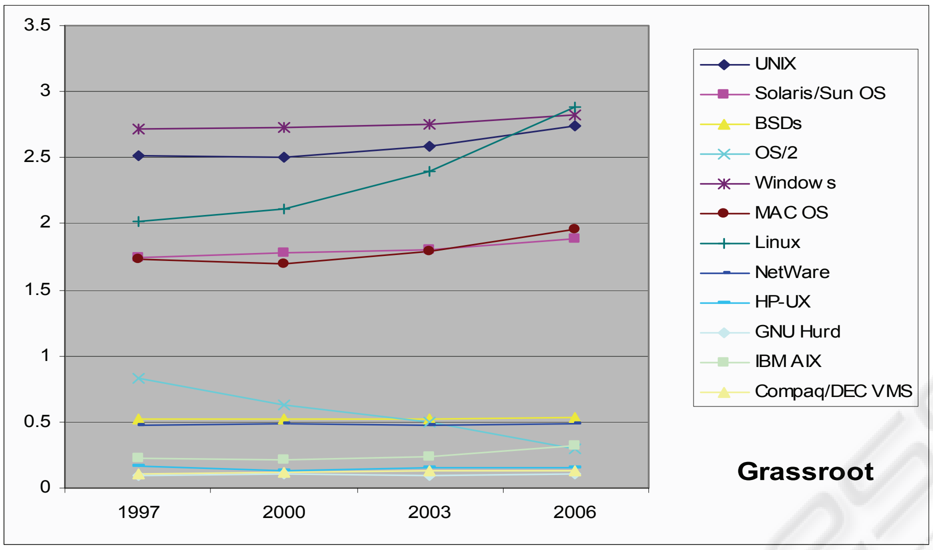

Figure 3: Evolution of Grassroots support for operating systems.

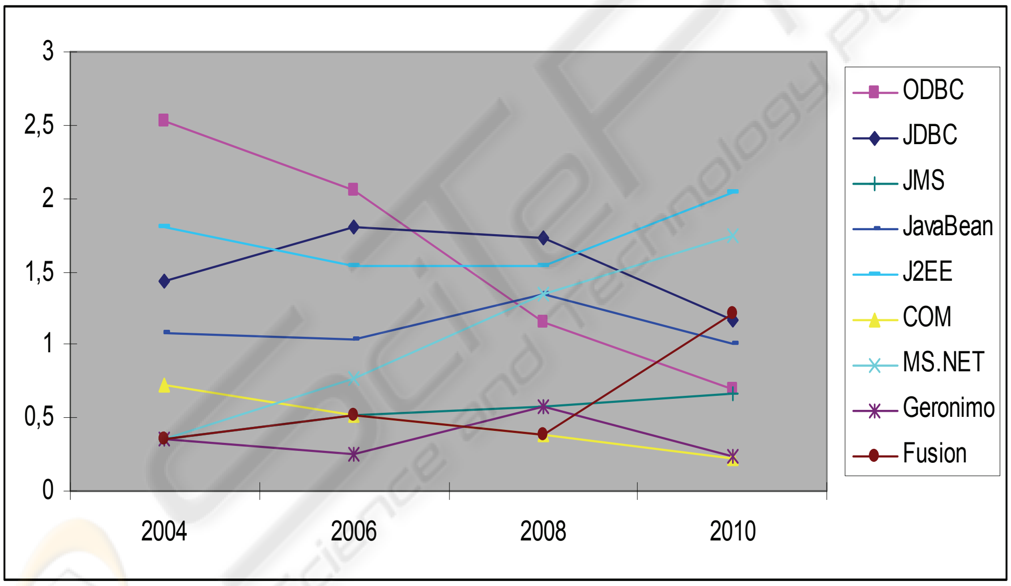

Figure 4: Evolution of institutional support from 2006 to 2010 (Actual and predicted).

3.3 Middleware Systems

To analyze the evolution of middleware systems, we have considered a sample of 18 middleware systems, including: ODBC, JDBC, JavaBean, EJB, COM, CORBA, Jini, JMS, MSMQ, MQSeries, MTS, .NET, J2EE, JBoss, Weblogic, Websphere, Geronimo and Fusion. 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).

4 A GENERIC EVOLUTIONARY MODEL

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.
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. 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).
5. 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.
6. 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://web.njit.edu/~mili/tecgeneric.xls.

4.2 Regression Model for Historical Trends

We model the state of success of a trend 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 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

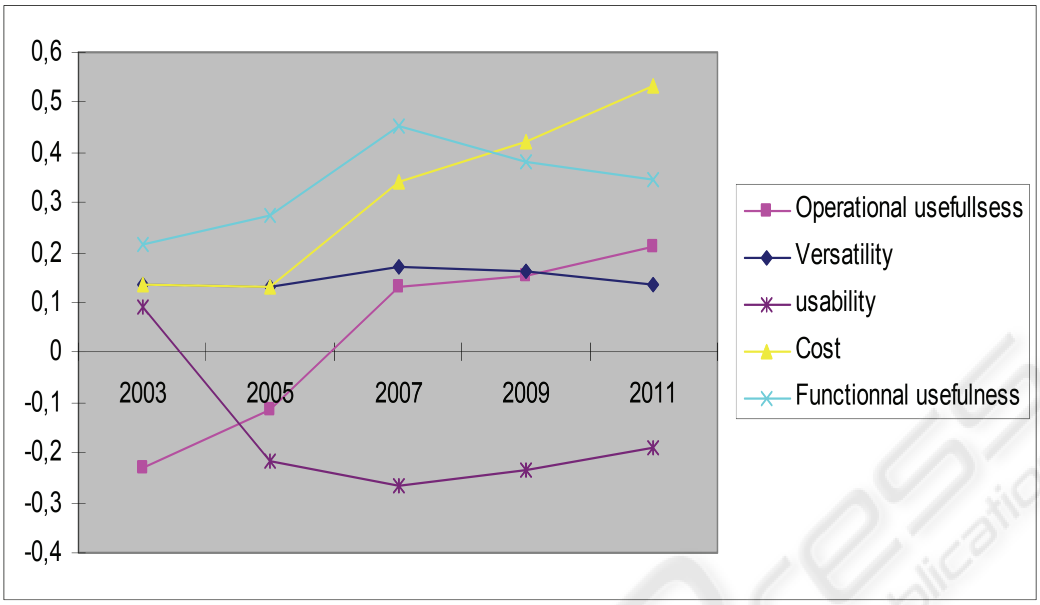

Figure 5: Relative importance of intrinsic factors for Grassroots support.

tool by feeding it past and present data and querying it on future data. Specifically, if we let *Ey* 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 α, β, γ, and δ that minimize the error term ε 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 α the parameter matrix for intrinsic factors, β the parameter matrix for extrinsic factors in 2009, γ the Parameter matrix for extrinsic factors in 2007, δ the parameter matrix for extrinsic factors in 2005 and ε an error term.

4.3 Profiling Successful Trends

When we study a single family of software trends (e.g. programming languages, operating systems, etc), it is useful to plot the chronological evolution of each member of the family; it may be useful to know what percentage of people will be using C++, or C three years from now. But if we are developing a generic evolutionary model, what is important is not the values of the extrinsic factors per se, but rather how intrinsic attributes are correlated to these values. For example, we may want to know the importance of usefulness, or usability, or versatility, or cost, for some extrinsic factor or another.
Consider the chart above, which plots the relative importance of the various intrinsic factors with respect to grassroots support, and how this evolves over time. From this chart we can infer for example that in 2011, success in terms of grassroots support is contingent upon the following criteria, ranked by order of decreasing importance: cost, functional usefulness, operational usefulness, versatility, and usability.

By contrast, to be successful in academia in 2011, a software technology trend needs to satisfy the following criteria, ranked by order of decreasing importance: cost, operational usefulness, versatility, functional usefulness, and usability.

Finally, to be successful in governmental quarters, a software technology trend must stress operational usefulness, followed by functional usefulness, followed by versatility, then cost, then usability.

5 CONCLUSIONS

In this paper we have collected the data and build a quantitative statistical model to capture the evolution of software technologies. This model can be

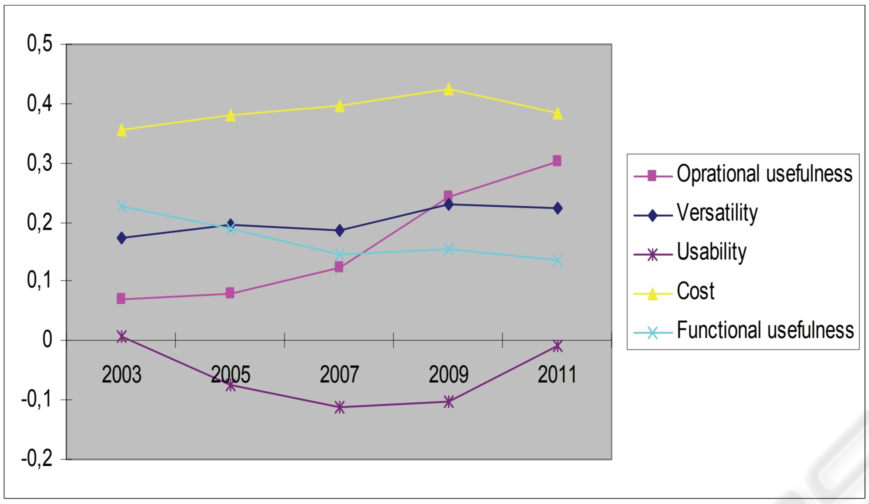

Figure 6: Relative importance of intrinsic factors for Institutional support.

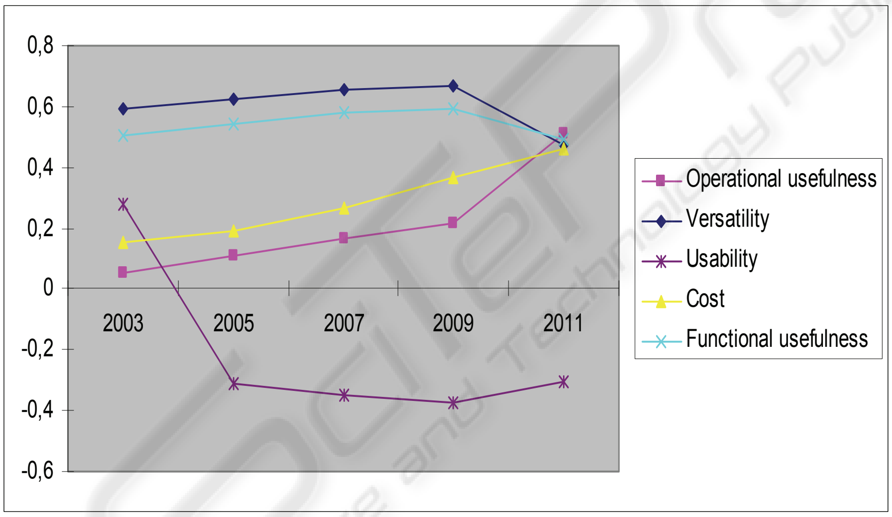

Figure 7: Relative importance of intrinsic factors for Governmental support.

applied to any software technology trend, as follows:

- We feed it information on intrinsic attributes of the trend.
- We feed it historical information on extrinsic attributes, such as past support from various relevant quarters.

Using this information, our proposed model can generate predictions in terms of future support from relevant quarters. Such a model can be used by various decision-makers as a source of quantitative information, to complement expert opinions and /or qualitative assessments. We are currently validating this model by preparing to apply it to a new empirical study, pertaining to data bases. One can have some level of confidence in its validity, on the basis of past validation of the individual empirical studies on which this model is based (programming languages, operating systems, middleware systems).

We hope the snapshot given in this paper reflects the potential that our data offers. The raw data that we have collected for this study is available online at http://web.njit.edu/~mili/tecgeneric.xls.

REFERENCES

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

Chen, Y. F., Dios, R., Mili, A., Wu, L. and Wang, K.F.

2005. Programming Language Trends: An Empirical Study. *IEEE Software.*

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.

Kuhn, Th. S. 1996. *Structure of Scientific Revolution.* University of Chicago Press.

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

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

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.