

# Qualitative Simulation and the Study of Software Evolution

Juan F Ramil                      Neil Smith

Computing Department  
Faculty of Maths and Computing  
The Open University

Walton Hall, Milton Keynes MK7 6AA, U.K.  
{j.f.ramil; n.smith}@open.ac.uk

## Abstract

*Empirical studies of software evolution will benefit from simulation techniques such as qualitative simulation, which addresses the sparseness of the empirical data and the imprecise knowledge about the relationships between the attributes being modeled. This paper introduces the technique and outlines qualitative simulation results of various high level models of the software evolution. It also shows how empirical data can be abstracted into a qualitative form, and the results compared to the qualitative simulation traces. The study concludes that the qualitative patterns observed in several different software systems studied display remarkable similarities and that similar patterns are produced by the models under specific external conditions. Results provide support for the models as planning aids and suggest that qualitative abstraction and simulation have great potential for further research in this area.*

## 1. Introduction

Continual fixing, adaptation and enhancement of real world software and their applications is needed as long as the software is used if stakeholders' satisfaction is to be maintained. This gives rise to the software evolution phenomenon [3, 7], whose importance has been recently widely acknowledged. Empirical studies of the phenomenon have led to the identification of commonalties in the evolution of E-type, that is, software addressing real world problems. The studies of Lehman and his associates over the years are of significance in this area [3, 7]. Their studies first led to three, now eight,

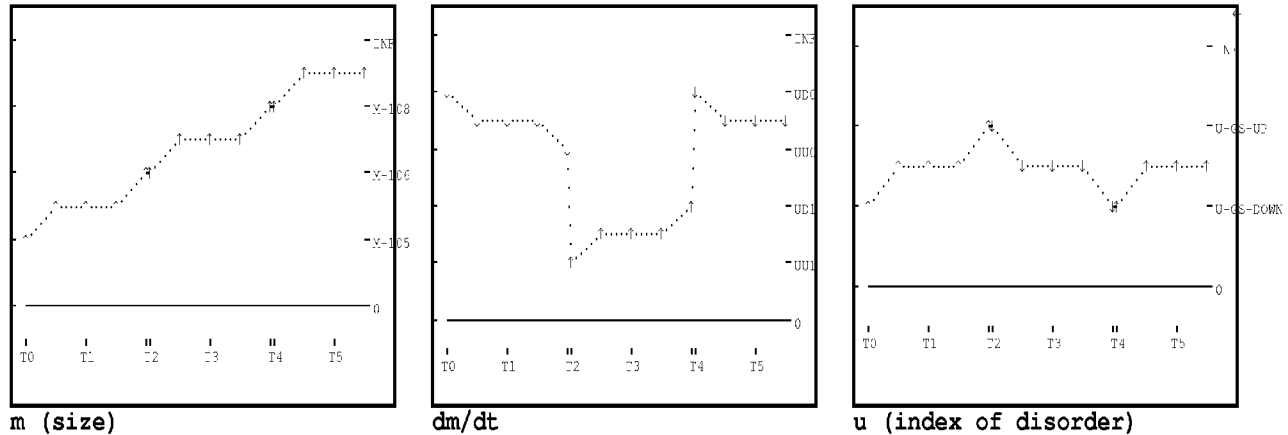
high level natural language statements, referred to as *laws of software evolution* [7]. Each of the statements is interpretable in terms of forces believed to be outside the immediate control of those directly involved in implementing the evolution of a software system. Interpretation also suggests that the behaviors implied in each of the statements are likely to emerge in spite of the technology used or method followed [7]. Over the years these views have justified the use of the term "laws" to refer to the statements. The latter have been and still are a topic of empirical research [3, 7, 8, 9, 12, 13]. Some of this work has involved simulation modeling. A number of quantitative simulation models based on the laws has been built, for purposes such as the modeling of functional growth in individual software systems over releases [3, 18], the study of long term impact of manpower allocation, release and complexity control policies on such growth [15, 20] and, more generally, in the investigation of mechanisms underpinning empirically observed behaviour and achieving an understanding of the phenomenon [4, 19].

All the above simulation work so far has faced an important challenge: the construction of a quantitative simulation model has required the specification of mathematical relationships between the attributes or variables involved in the model. If such relationships are not known precisely, the model builder has to make assumptions. The issue is critical when empirical data is scarce and imprecise knowledge about the relationships. However, qualitative reasoning techniques, such as qualitative simulation [6], illustrated by the model in figure 1 below, allow modeling at a higher level where such assumptions are not necessary. In recent papers [14, 16] the authors have described how such techniques can

Quantitative	Qualitative
$M(t) = M(t-1) + K_N \cdot f_E(U(t)) \cdot B_N(t)$ $U(t) = U(t-1) \cdot [1 + K_U \cdot f_M(M(t-1)) \cdot B_N(t) - K_S \cdot f_E(U(t-1)) \cdot B_S(t)]$ $f_E(U) = \exp(-U/K_E)$ $f_M(M) = M$	$B = \text{constant}$ $Dm = fE * Bn$ $B = Bn + Bs$ $fMBn = fM * Bn$ $Dm = d/dt m$ $fEBs = fE * Bs$ $Du = d/dt u$ $fMfE = fMBn - fEBs$ $FE = M-(u)$ $Du = u * fMfE$ $FM = M+(m)$

M: System size                      U: Index of disorder                       $K_U, K_S, K_N, K_E$ : Conversion and scaling factors  
 $f_E(U)$ : Relative efficiency of work     $f_M(M)$ : Impact of size on growth of disorder  
B,  $B_N, B_S$ : Total, progressive, and antiregressive budget or work force

**Figure 1: Quantitative and qualitative versions of the Woodside model**



**Figure 2: Woodside model with responsive management**

be applied to modeling long-term software evolution. This paper presents a summary of the concepts and results. The interested reader is referred to [14] for further details.

## 2. Qualitative simulation

In the QSIM simulation tool and formalism [6], a system is represented as a set of qualitative differential equations (QDEs), such as  $d/dt y = M^+(x)$ . Each QDE represents a large set of possible ordinary difference equations (ODEs) as each  $M^+$  (or  $M^-$ ) function represents the set of all monotonically increasing (or decreasing) functions. The values of variables are given relative to sets of qualitatively significant landmark values (such as zero and maximum budget). Simulation is then possible without precise knowledge of the functions involved and its output provides an overview of model's behaviours. The simulation traces (behaviours) retain the essential features while abstracting away unnecessary detail, such as constant scaling and conversion factor.

Qualitative simulation starts from a given initial state. The QSIM engine then generates all the possible, qualitatively distinct, successors of that state. Because the qualitative model encompasses many quantitative models, there are often several distinct successors for any given state. QSIM finds them all, and hence generates a tree of possible behaviours. For example, figure 1 shows

**Table 1: Some characteristics of the systems considered in this study**

System ID	Application Domain	Main Source Code Language	Technology of 1 <sup>st</sup> Release
OS/360-370	Mainframe OS	Assembly	1960s
A	Real Time	C	1980s
B	Real Time	C	1980s
C	Mainframe OS	S3 (*)	1970s
D	Information System	COBOL	1980s

(\*) similar to ALGOL

the Woodside model of software growth [20]; QSIM finds many behaviours for this model, including unbounded growth, stagnating growth, and growth in a series of distinct segments (figure 2). Kuipers [6] gives a fuller exposition on QSIM and how it operates.

Ramil & Smith [14] describe how QSIM can be applied to models of software evolution. The models explored in that paper [15, 16, 20] were selected because they are sufficiently complex that they cannot be solved algebraically. The paper describes how each of the models was converted to a qualitative form and discusses the different behaviours that resulted from their simulation. QSIM found all the behaviours described for the qualitative models and also discovered some additional behaviours which had not been described before. The paper also discusses the use of primitives to describe qualitative behaviour as illustrated by the trend description alphabet in table 2 below. Using the symbols in table 2, qualitative abstraction of empirically observed behaviour is presented in table 3. The simulation results

**Table 2: Pattern classification as behavioural primitives and their symbols**

$dx/dt$	$d^2x/dt^2$	Symbol
[+]	[+]	)
[+]	[0]	/
[+]	[-]	(
[0]	[+]	∪
[0]	[0]	—
[0]	[-]	∩
[-]	[+]	∪
[-]	[0]	∖
[-]	[-]	)
[+],[-] or [0]	[?]	?
[?]	[+],[-] or [0]	?

**Table 3: Summary of qualitative behaviours identified in empirical data**

System	Full Pattern	Simplified Pattern
OS-360/370	)/( )/?	) ( )
A	? ( )/(	( ) (
B	? ) ( )? /?	) ( )
C	( ) ( )/?	( ) ( )
D	( ) ( ) (	( ) (

are summarised in table 4. Further details are given in the next section.

### 3. Comparison with empirical data

As in the quantitative counterpart, in qualitative simulation one needs a systematic way of comparing simulation output to empirical data. In [14] we compared the theoretical results from the various models against some empirical data from five industrially evolved software systems (table 1). The data was collected in a previous research project [8]. The intent was to evaluate whether the models replicate the growth behaviour of such real systems in a qualitative sense. Before these data can be compared to the qualitative results from the models, the qualitative trends had to be identified in the empirical data.

The trend analysis approach in [14] extracts qualitative trend information from the data. It is an adaptation of the approach described in Colomer et al. [5]. In this approach, the signs of the first and second derivatives are used to characterize the qualitative trend over a period (table 2). Adjacent periods with the same qualitative trend are combined. The sparseness of the data meant that Colomer et al.'s procedure had to be modified. The modified procedure required two or more successive periods to have the same values for  $dx/dt$  and  $d^2x/dt^2$  before a trend is recorded; single period changes were ignored and do not generate trends. Additionally, sequences of alternating positive and negative values are taken to be zero. These abstraction rules led to the identification of more consistent patterns than the application of the original procedure [5]. The qualitative trend patterns for the five systems considered are

**Table 4: Summary of simulated behaviours**

Full Pattern	Simplified Pattern	Description
/	r	Unbounded 1
(—	(—*	Stagnated 1
) /	) /r	Unbounded 2
) / (—	) / (—	Stagnated 2
( ) ( ) ..	( ) ( ) ..	Unbounded 3
( ∩ ) \ ( ∪ ) / ( ...	( ) \ ( ) / ..	Oscillatory

presented in table 3. The table includes two columns: one with the full trend and the other with a simplified version of it in which any intermediate linear trend is ignored. With real measurements, the value of the differences (derivatives) is unlikely to be exactly zero. The question then is how close the derivative is to zero. The answer in many cases depends on the particular abstraction rules. This suggests that linear segments can be a by-product of the abstraction rules, as much as of the data. Until more robust abstraction rules are devised, the elimination of linear segments allows for a leaner and even higher level comparison with the simulation traces.

The qualitative simulation results are summarized as qualitative trend patterns in table 4, which includes, as in table 3, full and simplified patterns.

All the simplified empirical patterns in table 4 display a remarkable similarity suggesting that all can be reduced to a sequence of sigmoid growth curves that can be expressed as the pattern ( ) ( ). This empirical behaviour corresponds to the ‘unbounded 3’ simulated behaviour. Such behaviour has been generated by the three simulation models studied under the so called responsive management mode. Such changes in management policy can have a connection with the conceptualization of software evolution as a sequence of regular segments or stages separated by transitions [2, 13] and the ways in which requirements, features and their implementation evolve.

Many of the behaviours predicted by the models do not appear in the empirical data. The stagnated growth pattern may not appear simply because the data reflects systems being actively evolved. Other simulated growth patterns, such as when the complexity of the evolving software is driven (by means of for example anti-regressive [7] or refactoring work) to some low value and held there, do not appear to be in the empirical data. Further investigation is needed to determine if such behaviours are only possible in the modelling space or are ones that could appear in practice.

### 4 Related work

The vast majority of the software process simulation models addresses finer-grained planning and managerial issues [1] than those addressed here. With a similar focus, qualitative simulation has started to be applied to development projects [17]. The work presented here differs from these others in several aspects. This work focuses on searching for and explaining the triggers for commonalities in the patterns observed in long-term evolution across different systems, with the goal of achieving a greater understanding of the *E*-type evolution process.

This work has immediate connections with work by others. For example, the presence of unrecognizable periods, identified in table 2 by “[?]”, in the evolutionary trend of a particular software system can be linked to

discontinuities in the evolution of the system. Aoyama has discussed the presence of such discontinuities [2] and has suggested, for example, the use of architectural metrics to detect such a discontinuities. The behavioural abstraction method used to derive the patterns in table 4 from empirical data offers an alternative way of determining the presence of such discontinuities. Discontinuities can also be linked to the recently proposed view that different stages can be identified in the life-cycle of software systems [11]. It is likely that transitions from one qualitative pattern to another in an evolutionary attribute can be associated with transitions from one such stage to another.

## 5. Further work

The work presented in this paper can be extended in a number of ways. Additional work is needed in order to identify and assess robust procedures for extracting qualitative trends from sparse data and that are appropriate to the type of data available in the software engineering domain. Given that different abstraction procedures may lead to differences in the patterns observed, the further investigation of these techniques and the issue of linear segments discussed above is important to put this line of investigation on a firmer foundation.

Another avenue of research is to refine the models of the software evolution process. For instance, all the models discussed here assume that the inputs to the development process (work force, budget, effort) are constant whilst a system is evolved. However, the empirical data suggest that the effort applied tends to vary between segments of the system lifetime [9, 13]. This issue complicates the process of comparing the models' behaviours with the empirical data. The models can also be extended by taking a "trend and ripple" view of the growth process [3], similar the qualitative simulation technique of abstraction by time scale [6]. Finally, qualitative simulation can take place of different models to the ones so far proposed and studied.

As discussed in [12], qualitative simulation offers a number of advantages over qualitative simulation in addressing issues such as the empirical support for the laws of software evolution. The technique can advance this line of research and, in general, the research towards the formation, from the laws and other elements, of a formal theory [9].

## 6. Conclusions

Qualitative simulation is a powerful technique for creating and using models in the presence of incomplete information. The abstraction of the model to consider only its qualitatively distinct behaviours is appropriate considering the sparseness of the empirical data. Existing quantitative models can easily be abstracted into QDEs

and qualitative simulation results agree with quantitative results. The validity of these models can also be assessed by comparing them to empirical data, which requires that the key qualitative features in those data be identified. Qualitative trend abstraction assists in this difficult task and produces a concise summary of the trends in the data. This abstraction also focuses attention on the interesting points in a system's behaviour; for example, by identifying the locations of any discontinuities in the growth trends and where the patterns change. Finally, the comparison between the empirical data and the behaviours predicted by the models shows that the models, and hence the theories they embody, represent only a partial understanding of the process at work in software evolution. The tools of qualitative simulation and qualitative trend comparison represent a powerful technique for developing and evaluating theoretical advances in this area.

Research into all these processes (including Lehman's laws of software evolution) can be seen as falling into one of two groups. The first line of research starts with empirical data and asks both how it can be generalized into laws or models, and how well the empirical data supports such generalization [12]. Another line of research takes the generalizations as starting point and seeks to derive practical results such as guidelines for management [8]. The techniques presented here can be of help in both lines of investigation by providing a fruitful intermediate level between high-level empirical generalizations and conventional quantitative simulation.

## 7. References

- [1] Abdel-Hamid T.K. and S.E. Madnick *Software Project Dynamics—An Integrated Approach*, Prentice Hall, Englewood Cliffs, NJ, 1991.
- [2] M. Aoyama "Metrics and Analysis of Software Architecture Evolution with Discontinuity", *Proceedings of the Fifth International Workshop on Principles of Software Evolution (IWPSE 2002)*, 2002, pp. 103 – 107.
- [3] L. Belady and M.M. Lehman "A Model of Large Program Development", *IBM System Journal* **15**(3), 1976, 225–252.
- [4] B.W. Chatters, M.M. Lehman, J.F. Ramil, and P. Wernick "Modelling a Long Term Software Evolution Process", *Software Process—Improvement and Practice*, **5**(2/3), 2002, 91–102.
- [5] J. Colomer, J. Melendez, and F.L. Gamero "Qualitative Representation of Process Trends for Situation Assessment Based on Cases", *Proc. of the 16th International Workshop on Qualitative Reasoning (QR2002)*, 2002, pp. 37 – 43.
- [6] Kuipers, B. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*, Cambridge, Massachusetts: MIT Press, 1994.
- [7] Lehman, M.M. and L. Belady, *Software Evolution — Processes of Software Change*, Academic Press, London, 1985.
- [8] M.M. Lehman and J.F. Ramil "Rules and Tools for Software Evolution Planning and Management", *Annals of Software Engineering* **11**(1), 2001, 15 – 44

- [9] M.M. Lehman and J.F. Ramil "An Overview of Some Lessons Learnt in FEAST", *Proceedings, WESS 2002*, pp. 41 – 47
- [10] M.M. Lehman, J.F. Ramil, and U. Sandler "An Approach to Modelling Long-Term Growth Trends in Software Systems", *Proceedings ICSM 2001*, pp. 219 – 228
- [11] V.T. Rajlich and K.H. Bennett "A Staged Model for the Software Life Cycle", *Computer*, July 2000, 66 – 71
- [12] J.F. Ramil "Laws of Software Evolution and Their Empirical Support", panel statement, *Proceedings, ICSM 2002*, 71 – 71
- [13] J.F. Ramil *Continual Resource Estimation for Evolving Software*, PhD dissertation, Dept. of Computing, Imperial College, London, 2003
- [14] J.F. Ramil & N. Smith "Qualitative simulation of models of software evolution", *Software process improvement and practice*, 2003, to appear.
- [15] J.S. Riordan "An Evolution Dynamics Model of Software Systems Development", *Software Phenomenology - Working Papers of the (First) SLCM Workshop, Pub ISRAD/AIRMICS, Comp. Sys. Comm. US Army, Fort Belvoir VI*, 1977, 339 – 360
- [16] N. Smith and J.F. Ramil (2002) "Qualitative Simulation of Software Evolution Processes", *Proceedings, WESS 2002*, pp. 41 – 47
- [17] A.J. Suarez, P.J. Abad, R.M. Gasca, and J.A. Ortega (2002) "Qualitative Simulation of Human Resources Subsystem in Software Development Projects", *Proceedings of the 16th International Workshop on Qualitative Reasoning*, 2002, pp. 169 – 176
- [18] W.M. Turski "The Reference Model for Smooth Growth of Software Systems Revisited", *IEEE Transactions of Software Engineering*, **28**(8), 2002, 814 – 815
- [19] P. Wernick P and M.M. Lehman "Software Process White Box Modelling for FEAST/1", *Journal of Systems and Software*, **46**(2/3), 1999, 193 – 201
- [20] C.M. Woodside "A Mathematical Model for the Evolution of Software", *CCD, Imperial College, Research Report 79/55*, Apr. 1979, also in *Journal of Systems and Software*, **1**(4) (1980), 337 – 345.