A System Dynamics Model of Software Evolution
(revised version of charts 15 May 2001)

University of Sannio
3 May 2001

Juan F Ramil
(presenting a modelling work undertaken jointly with Dr Goel Kahen)

Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
tel. +44 (0) 20 7594 8216
fax. +44 (0) 20 7594 8215

ramil@doc.ic.ac.uk
http://www.doc.ic.ac.uk/~ramil
Objectives

To introduce the application of system dynamics simulation models as a tool to plan, manage and control software evolution processes

- To provide an example of evolution process model developed in FEAST

- To illustrate the top-down modelling approach followed in FEAST, which is based on successive refinement starting with a simplified high-level process view
Early Models of Evolution Dynamics

- Study of OS/360 and other systems during late 60s and early 70s led to
dynamic models of program growth such as:
  Belady and Lehman 1972,75; Riordan 1977 and Woodside 1979

- Such models replicated observed growth trends over time and releases
  suggested management guidelines

- In late 80s and early 90s that dynamic modelling of software processes gains wider interest
  but with focus on ab initio development
  - triggered, at least in part, by Abdel-Hamid’s book on Software Project Dynamics

- Not wide interest in dynamic modelling of long-term evolution, with a few exceptions

- This contrasts with the fact that evolution - not ab initio development - consumes largest
  portion of effort in software organisations

- Process models developed in FEAST project (http://www.dse.doc.ic.ac.uk/~mml.feast)
  address situation by focusing on long-term evolution
  - models are generally consistent with observations in laws of software evolution
  - reflect input from industrial collaborators
Evolution: Closing the Loop

- **Installation**, introduction into *usage* closes major **feedback loop**
- Driver of **continuing** software system **evolution**

This loop is starting point for process modelling in FEAST
Several classifications of maintenance and evolution activity have been proposed over the years.
- Focus has been on types such as fixing, adaptation, and enhancement.

Lehman’s proposal (1974) includes 2 classes of activity (or effort):

- **Progressive** activity - directed to achieve increase functionality, better performance and in general to change the behaviour of the software as perceived by users.

- **Anti-regressive** activity - undertaken to control complexity and its growth and in principle not altering properties of software as experience by users.

  Examples of anti-regressive work:
  - restructuring, rewriting, re-engineering, re-documentation, removing dead code or duplicates, *refactoring*, moving to components and/or higher level languages, etc, etc.

Basis for a dynamic model exploring role of increasing complexity.
A Model of Lehman’s 2nd law - Increasing Complexity

- **Complexity** seen as anti-regressive work deficit - a process metrics, becomes operational definition of software complexity

- Some assumptions:

  Total_effort (e.g. person-hours per month) = Progressive_effort + Anti_Regressive_effort
  Anti-Regressive Effort = Total Effort * Anti_Regressive_Work_Fraction
  \[ 0 \leq \text{Anti_Regressive}_\text{Work}_\text{Fraction} \leq 1.0 \]

- **Insufficient** anti-regressive work will lead to complexity increase

- **Excessive** anti-regressive work will lead to resource waste

- **Optimum** value for \text{Anti_Regressive}_\text{Work}_\text{Fraction} somewhere in-between

  A system dynamic model to explore this issue follows
Base Model

Exogenous work requests → Work Waiting

Work Waiting → Implementation

Implementation → Work Implemented
Endogenous work request = delay3(USER CHANGE REACTION MULTIPLIER * Release, USER CHANGE REACTION DELAY)

Vensim provides several functions, such as delay3, to simulate dynamic effects. See reference manual for details. This equation is a starting point to model the impact of the release work in the usage domain. It awaits empirical validation - in the simulation results presented later, the actual value of Work_Waiting has no impact.
Individual Productivity vs Team Size - A Proposed Formulation

Following Abdel-Hamid and Madnick:
Productivity = Potential_Productivity*Communication_Losses

Communication Losses

• In a group of size \( n \), number of communication channels is \( n(n-1)/2 \)

Assuming that the loss per communication link is constant impact on software development rate of team can be modelled, for example, by function that decreases as the size of the team increases:

\[
\text{Communication Losses in percent} = 1 - \frac{k(n-1)n}{100} \\
\approx 1 - \frac{k \cdot n^2}{100}
\]

where \( n \) is the team size (number of people in the team) and \( k \) is a suitable constant, obtained from empirical data

Abdel-Hamid and Madnick, for example, have suggested that for a team of 30 people, 50 percent of the effort is subsumed by communication activity and hence unavailable for software development - this observations leads to \( k = 0.06 \)
Productivity vs Team Size - A Proposed Formulation

Productivity = Potential Productivity*Communication_Losses

Potential Productivity

• Many factors can be considered here - e.g. motivation, learning

• We refer to Abdel-Hamid and Madnick’s book for detailed discussion

• Consider here only one aspect, the increase of individual productivity due to build-up of the experience base as a team grows in size - we refer to it here as synergy

Assuming that the gain in potential productivity due to synergy can be modelled, for example, by function that increases as the size of the team increases:

\[ \text{Synergy} = \{a + b(n/n+c)\} \]

where \( n \) is the team size (number of people in the team) and \( a, b \) and \( c \) are suitably selected constants, obtained from empirical data

• Many factors can be considered here
• Behaviour is qualitatively as expected - quantitative detail will certainly vary from project to project and organisation to organisation

• Of course, one will need to be calibrate behaviour to real data - for the moment, an starting point
Third Refinement

Productivity = NOMINAL PRODUCTIVITY * (0.67 + (0.7 * TEAM SIZE / (TEAM SIZE + 0.7))) * max(0, 1 - (0.0006 * TEAM SIZE * (TEAM SIZE - 1)))

Implementation = Progressive Effort * Productivity

Progressive Effort = TEAM SIZE
Productivity vs Anti-Regressive Deficit - A Proposed Formulation

Anti_regressive_deficit = Cumulative_Work_Implemented - Cumulative_Anti_regressive_Activity

• What is the relationship between Productivity_loss and Anti_regressive_deficit?

• One may expect that productivity loss increases as Anti_regressive_deficit grows

• If Anti_regressive_deficit < 0 then
  Productivity_loss = 0
otherwise
  Productivity_loss = f(Anti_regressive_deficit)

• f() defined using Vensim “Look Up Table” feature

• Of course, one would need to be calibrated to real data - for the moment, an starting point
Fourth Refinement

Productivity = NOMINAL PRODUCTIVITY * 
(0.67 + (0.7 * TEAM SIZE / (TEAM SIZE + 0.7))) * MAX(0,1 - (0.0006 * TEAM SIZE * (TEAM SIZE - 1)))

*Productivity Loss

Progressive Effort = (TEAM SIZE - Anti regressive effort)

Anti regressive effort = TEAM SIZE * ANTI REGRESSIVE WORK FRACTION
Example of Model Output

- Exploring consequences of various values \textit{Anti\_Regressive\_Work\_Fraction} (arwf)

- Rate of functionality released to users
  - a possible indicator of evolution \textit{success}
  - represented in model by \textit{Cumulative Fielded Functionality} \textit{CMF}

- Models enables to study effects of different allocation of resources, e.g.
  \textbf{arwf 0 pc}: no resources allocated to anti regressive work
  \textbf{arwf 40 pc}: 40 percent of resources allocated to anti regressive work
  \textbf{arwf 60 pc}: 60 percent of resources allocated to anti regressive work
An Example of Model’s Output (cont.)

- **Anti_Regressive_Work_Fraction** (arwf) = 40 percent provides largest growth after 200 months

- **Anti_Regressive_Work_Fraction** (arwf) = 0 percent maximises short-term behaviour

![Graph for Cumulative Work Released](image)

- Simulation results support the **convenience** of an appropriate level of **anti-regressive** activity in a software process

Next step, model refinement to reflect higher level of process detail
Further extended and refined model

- **Work Identified**
- **Work Accepted**
- **Work Prepared for Implementation**
- **In Progress**
- **Work Ready to Release**
- **Work Implemented**
- **Submission Flow**
- **Preparation Flow**
- **Validation and Integration Effort Release Policy**
- **Progressive Effort**
- **Preparation Effort**
- **TEAM SIZE**
- **NORMAL PRODUCTIVITY**
- **SYSTEM TYPE MULTIPLIER**
- **Cumulative Fielded Functionality**
- **Release Policy**
- **TIME STEP**
- **INTEGRATION PRODUCTIVITY FACTOR**
- **PROGRESSIVE FRACTION**

Additional notes:
- Other additions and changes identified
- Change plus defect discovery factor
- Requirement Change flow
- Demand obsolescence
- Fielded Functionality Satisfying Current Needs
- Software release
- Integration flow

Related concepts:
- ACCEPTED TARGET
- PREPARATION PRODUCTIVITY FACTOR
- PREPARATION EFFORT MULTIPLIER
- PROGRESSIVE EFFORT MULTIPLIER

15 May 2001 (c) All rights reserved. Juan F. Ramil
Some Modelling Tips

• **Avoid**, if and when possible, construction of very **large models**
  - large models involve risk of getting out of intellectual control

• One needs externally imposed **discipline**
  - identify sub-systems in the model
  - refine step-by-step as more detail needs to be reflected in model
  - achieve model realism with the smallest possible number of variables and relationships

• Quantification of **soft factors** - not straight forward
  - examples of soft factors: work pressure, motivation, knowledge, expertise
  - look for what has change or is likely to change - constant aspects can be in general factored out from the model

• **Warning** - Large differences in quantitative detail may be expected from process to process
  Thus, **a model must be calibrated to real data** reflecting a particular software evolution process **before** its use as policy assessment tool

• Start **data collection as soon as possible**, making use of cheap, available records, first
  - data collection is expensive and must be well justified
  - modelling unprecedented situations and/or without reference modes is difficult
Simulation, Process Improvement and Process Maturity Level

- Process maturity
  - higher maturity presumes higher degree of repeatability, improved performance

- Example: CMM - Capability Maturity Model, developed at SEI, CMU
  - considers level 1 - less mature- to level 5 - most mature processes

- Christie argues that simulation can play a role at *any* level

For further details see:
Role of Process Modelling - a Caveat

- According to Tom DeMarco - in a recent presentation at Imperial College - these are some of the skills that a manager in a software organisation requires - note that most of them are people skills, such as
  - hiring
  - thanksgiving
  - praising
  - conflict resolution
  - team work - in particular, works as a team with other managers
  - product and process management - e.g. simulation techniques

- Simulation models, metrics, estimation techniques offer only a tool
  - Barry Boehm recommends to rely on more than one model or tool
  - Our view is that use of process models should nor lead to a mechanistic view of process neither diminish human role, but understand implications of policies, mitigate risks