THE IRISH SOFTWARE
ENGINEERING RESEARCH CENTRE

Investigating the Gap between Quantitative and Qualitative/Semi-quantitative Software Process Simulation Models: An Explorative Study

He (Jason) Zhang

ICSP 2009, MAY 16-17, Vancouver, Canada

Outline

- *Background and motivations*
- *Model transformation*
  - *Diagramming, Level & rate, Delay*
- *Reference software process*
  - *Hypotheses in software evolution*
  - *Simplified quantitative model*
- *Corresponding qualitative/semi-quantitative model*
- *Simulation and comparison*
  - *Results from qualitative & semi-quantitative simulation*
- *Conclusions*

Background and motivations

- **State-of-the-practice of SPSM**
  - Software Process Simulation Modeling (SPSM): an effective approach (using simulation) to help evaluate and manage changes made to software projects and organizations.
  - In 2007, we undertook a systematic literature review (SLR) of SPSM research in the past decade (1998-2007).
  - Most of process models are quantitative, and rely on reliable history data, which limits SPSM adoption in practice.
  - System Dynamics are the most used modeling paradigm in SPSM research.
  - Most simulation models were also one-off developed, model reuse was often omitted by many SPSM studies.
  - The previous work explored qualitative aspects of simulation modeling of software process at different process scales.

Background and motivations

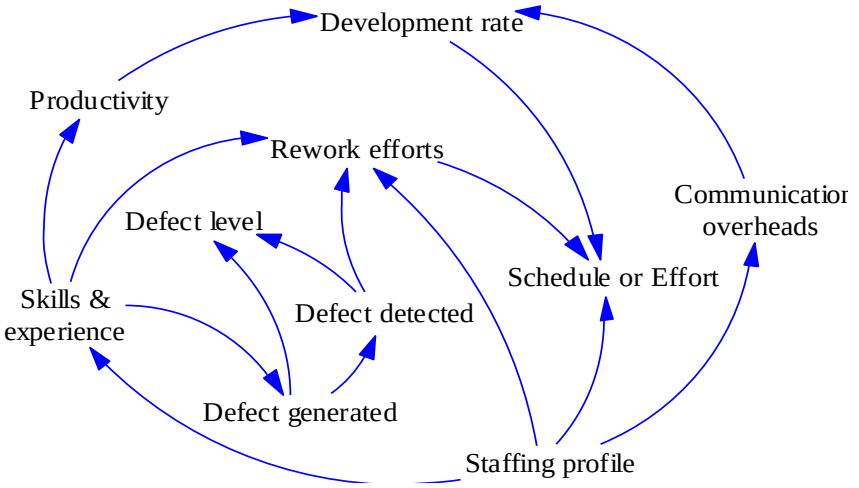
- **Research questions**
  - Does structural equivalency exist between two types of simulation modelling, i.e. qualitative vs. quantitative?
  - Are they capable of providing different zooms to software process research?
  - Is it possible to develop one process model for both quantitative and qualitative simulation studies?
  - Are modellers or users able to choose either type of simulation in terms of data availability and quality?
  - If so, how to transform a process model between quantitative and qualitative (semi-quantitative) simulation without missing structural information?
  - Explorative study needs to model a typical software process using the most popular simulation paradigm.

Background and motivations

- **A typical software process**
  - Based on the evidence derived from our systematic literature review of SPSM research again, software evolution is a typical software process, one of the most modeled process scales and topics in the decade (’98 – ’07).
  - Successfully transforming simulation model of software evolution may provide effective proof for answering RQs.
- **Reference model selection**
  - Paradigm: system dynamics (SD) are the most used simulation paradigm (out of 10) for SPSM in the decade.
  - Complexity: the reference model contains common elements and relations of SD modeling, but not complicated much.
  - Summary: a clear, simplified system dynamics model of software evolution process as reference in this study.

Model Conversion

- **Diagramming**
  - Causal loop diagram (CLD) used in SD modeling represents interdependencies (with arrowed connections) and feedback processes (with + or – polarity).
  - Abstract structure diagram (ASD) used in qualitative simulation (QSIM) provides more explicit notations and links, such as sum and product identifiers, monotonic functions.
  - A CLD can be converted into its corresponding ASD, but needs more explicit and specific qualitative assumptions.



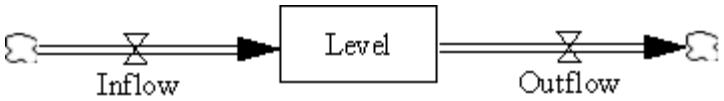
Model Conversion

- **Level and rate**
  - In SD, *levels* are absorbing inflow *rate*, and accumulating the difference (net flow) between the inflow and outflow *rate*.

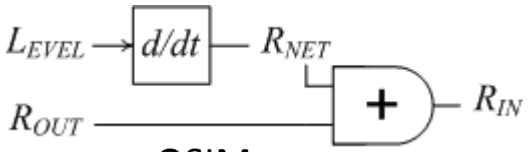
$$Level\,(t) = \int_{t_0}^{t} [inflow\,(s) - outflow\,(s)]ds + Level\,(t_0)$$

  - Instead, QSIM models systems using differential relations, the net flow (rate difference) is not explicitly presented.

$$\frac{\partial}{\partial t}\,(Level) = inflow\,(t) - outflow\,(t)$$



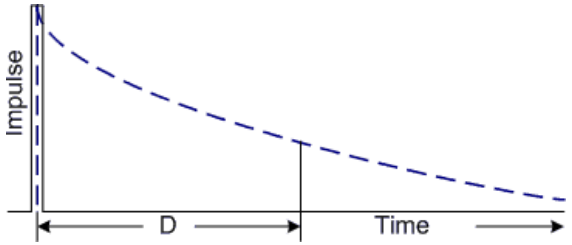
SD

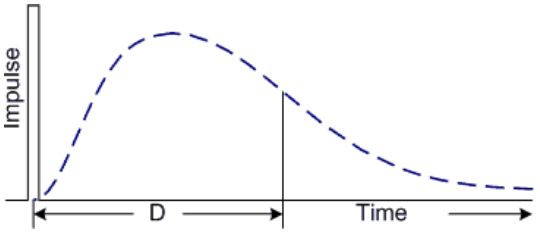


QSIM

Model Conversion

- **Delay**
  - A *delay* is determined by the length of average delay (time) and its transient response.
  - Exponential delay is the most used delay class in SD, in which 1st- and 3rd-order delays are common types in SPSM.

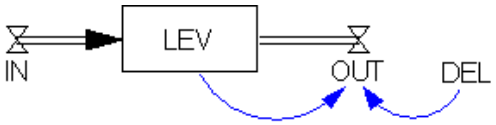




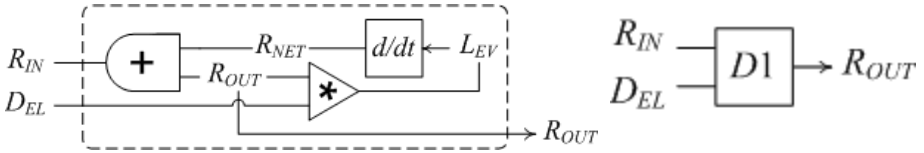
  - As time is described qualitatively in QSIM, it is not necessary to handle a delay with *qualitative* length (this transformation can be implemented in semi-quantitative simulation).

Model Conversion

- **Delay**
  - Implement first-order delay in semi-quantitative simulation:

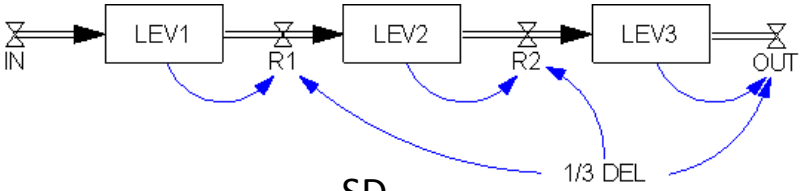


SD

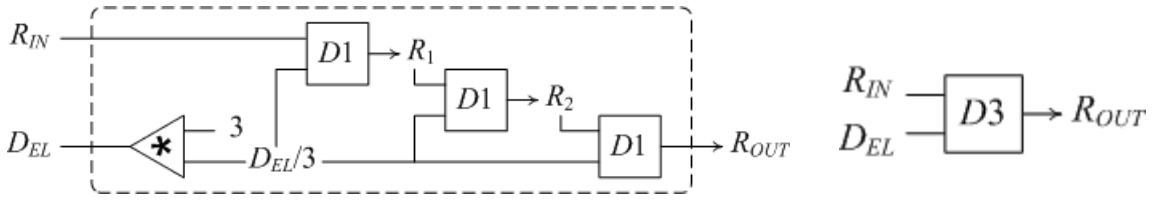


QSIM

  - Implement third-order delay in semi-quantitative simulation:



SD



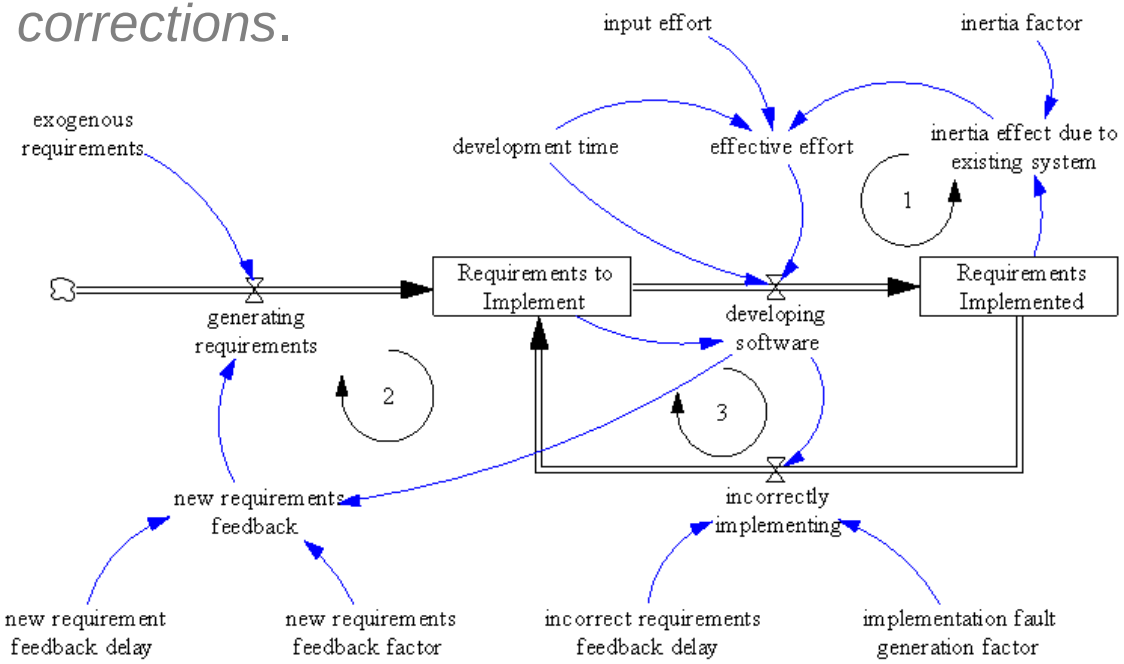
QSIM

Reference software process

- **Software evolution process**
  - Software evolution is a disciplined phenomenon, basically, four important feedback hypotheses are identified by previous related studies (in SLR):
  1. ***Inertia-like (anti-regressive) effect* due to system growth**: increasing the size of software and changes in unanticipated directions reduce its enhancement and modification.
  2. ***Decreasing effect* of knowledge coverage**: increasing complexity of software reduces people's ability to change it.
  3. ***Progressive effect* due to new requirements**: the new functionalities of upgraded software enable users to exploit more opportunities of system use, results in new demand.
  4. ***Correction* of fault implementation**: a small portion of units is gradually perceived not suitable, needs further correction.

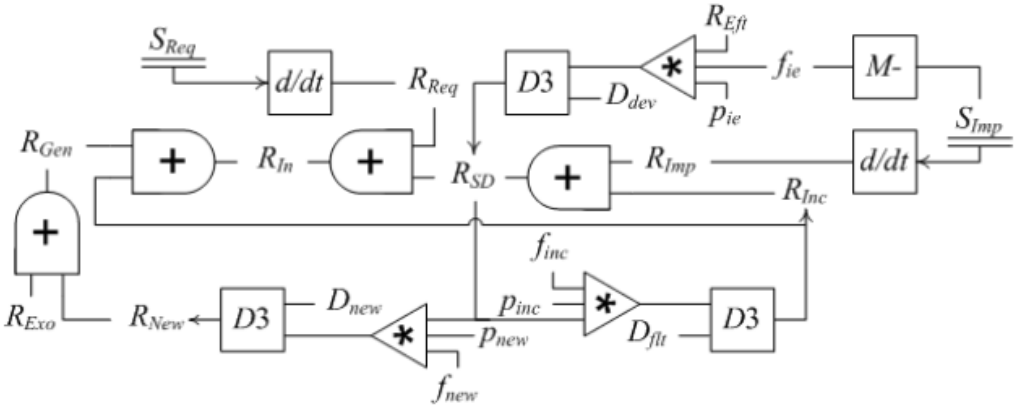
Reference software process

- **Simplified quantitative model**
  - Reference quantitative model is based on several SD models of software evolution published in the last decade.
  - It is simplified to be single-module, but includes 3 typical feedback loops just mentioned: *anti-regressive*, *progressive effect*, and *corrections*.



Corresponding qualitative model

- **Qualitative modeling**
  - SD model of software evolution is converted into QSIM (ASD) for qualitative simulation based on the scheme developed.
  - Qualitative model with semi-quantitative extension:



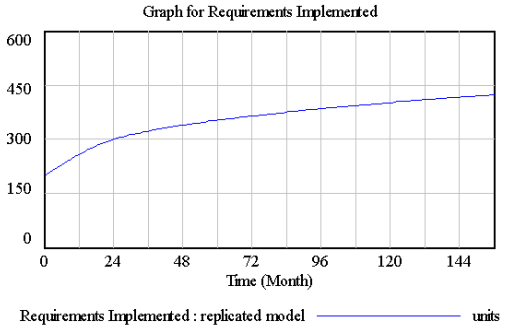
SReq: Requirements to implement
RReq: Requirement generation rate
RNew: New requirement feedback rate
RSD: Software development rate
RIn: Requirement input rate
REft: Effective effort rate
f_{new}: new requirement feedback factor
Ddev: Development delay time
Dinc: Incorrect requirement feedback delay
p_{new}: new requirement feedback policy factor
SImp: Requirements implemented
RImp: Requirement implementation rate
RInc: Incorrect implementation rate
RGen: Requirement generation rate
RExo: Exogenous requirement rate
f_{ie}: inertia factor
f_{inc}: fault generation factor
Dnew: New requirement feedback delay
p_{ie}: inertia scaling policy factor
p_{inc}: implementation fault feedback policy factor

Corresponding qualitative model

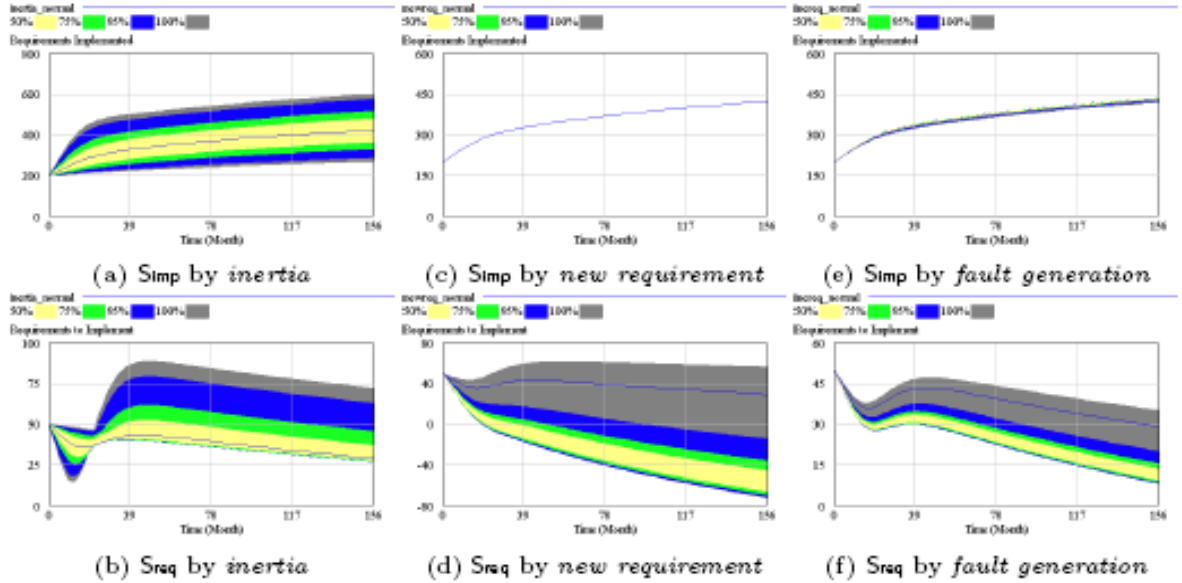
- **Qualitative assumptions**
  - Based upon the converted qualitative model, the inherent 9 qualitative assumptions of reference model can be extracted
    - The *incorrectly implemented requirements*, as a small portion of r*equirements implemented* is returned to *requirements to implement* for rework;
    - *Requirements to implement* come from *exogenous requirements*, *new requirements feedback* and *incorrect requirements feedback;*
    - Increasing *existing system size* incurs more effort needed for `*anti-regressive activities*', and decreases *software development rate*;
    - There is no exogenous requirements (Rexo = 0) during evolution process;
    - ……

Simulation and comparison

- **Quantitative simulation**
  - System size over time simulated by the simplified quantitative model, it terminates at the 156th month.

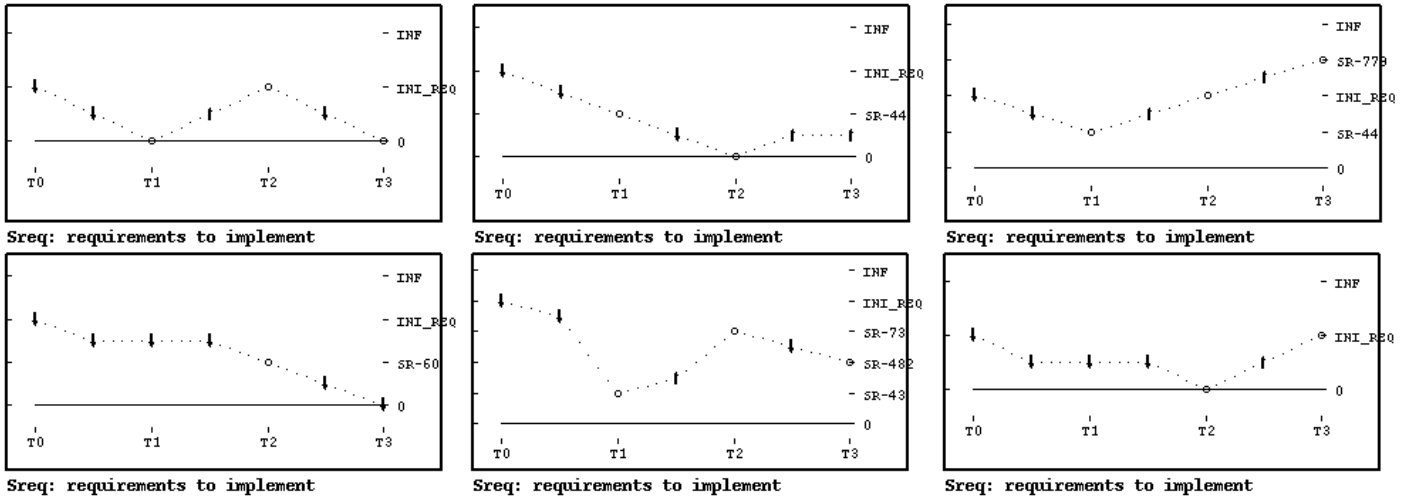


  - Three policy factors are selected for sensitivity analysis .



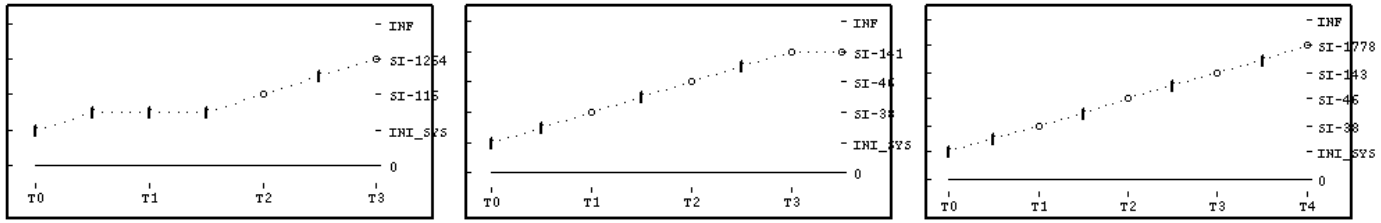
Simulation and comparison

- **Qualitative simulation**
  - As time is qualitative in QSIM, simulation is set to terminate after generating a large number of behaviours, which is sufficient to observe patterns.
  - **Requirements to implement** (Sreq) may gradually drop (down to zero), then rebounds to a certain level.



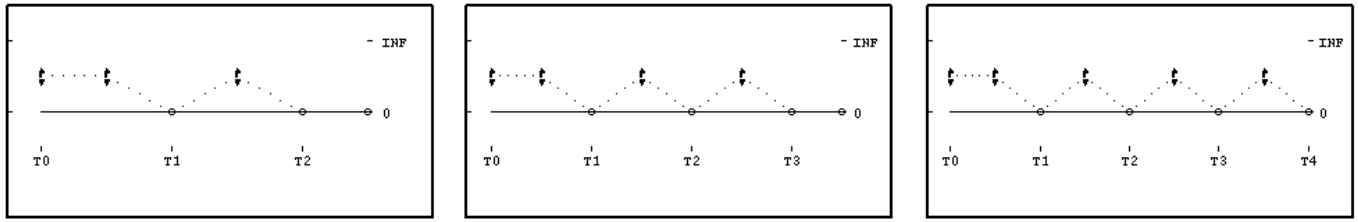
Simulation and comparison

- **Qualitative simulation**
  - **Requirements implemented** (Simp) presents growing trend at all time.



Simp: requirements implemented | Simp: requirements implemented | Simp: requirements implemented

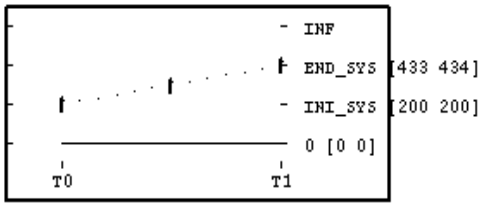
  - **Requirement implementation rate** (Rimp) oscillates over time, may reach equilibrium state after several oscillations or more.



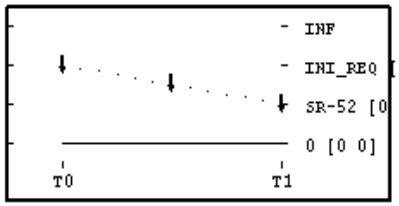
Rimp: net implement rate | Rimp: net implement rate | Rimp: net implement rate

Simulation and comparison

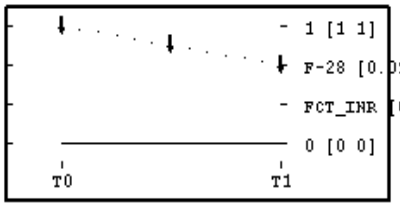
- **Semi-quantitative simulation**
  - **Single-point value simulation** Q2 algorithm generates two behaviors consistent with the reference model, which exactly terminate at the 156th month.



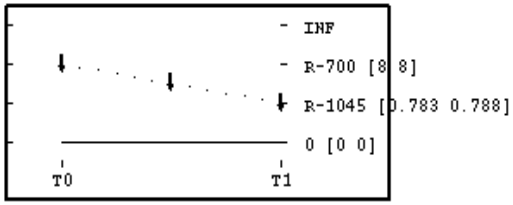
Simp: requirements implemented



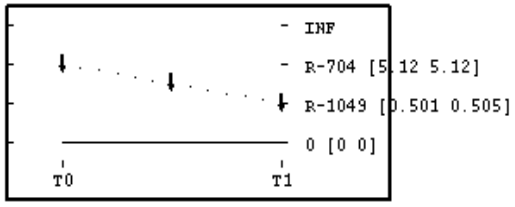
Sreq: requirements to implement



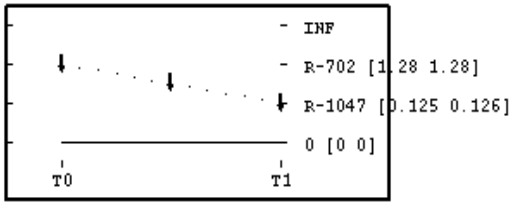
fie: inertia effect factor



Rsd: software develop rate



Rnew: new requirement rate



Rinc: incorrect implement rate

Simulation and comparison

- **Semi-quantitative simulation**
  - **Value-range simulation** uses Q2 and Q3 algorithm to reflect system growth due to uncertainty on three quality factors.

	SImp by f_{ie}	SImp by RNew	SImp by RInc
SD	[270, 601]	[434, 434]	[424, 443]
SQSIM (Q2)	[220, 627]	[423, 525]	[410, 527]
SQSIM (Q3)	[263, 610]	[430, 455]	[420, 471]

  - The difference between the two approaches may be caused by 1) the unique simulation mechanism by behavior chattering, 2) the missing points in SD due to sampling and probability distribution, 3) *step-size* in Q3 has not been completely optimized.

Simulation and comparison

- **Comparison**
  - **For modeling**, compared to causal loop diagram, qualitative modeling provides a more rigorous approach by starting at explicit assumptions and applying more specific constraints;
  - CLD model does not offer simulation capability, QSIM does;
  - They both can be quantified to become quantitative models.

  - **For simulation**, quantitative model can reflect variable's varying trend with more details than qualitative model;
  - When dealing with uncertainty, probability associated with value range is also required for quantitative simulation, whereas qualitative (and semi-quantitative) simulation allows lack of such information.

Conclusions

- This study enables the possibility of model reuse between quantitative and qualitative process simulation, and provides modelers an alternative simulation approach without creating a new model from scratch.
- An element level mapping from SD model to qualitative/semi-quantitative modeling (ASD) is established, and vice versa.
- A model transformation scheme from a quantitative model to qualitative model is implemented by using the element mapping. Given a qualitative model and its quantification, it is possible to covert construct its corresponding quantitative (SD) model.
- The software evolution processes are revisited using qualitative modeling and simulation.
- The modeling process and characteristics, and further the comparison of simulation capability and results between the two approaches are presented.

THE IRISH SOFTWARE
ENGINEERING RESEARCH CENTRE

Thank you!