

RVSim: A Simulation Approach to Predict the Impact of Requirements Volatility on Software Project Plans

Dapeng Liu, Qing Wang, Junchao Xiao, Juan Li, Huaizhang Li

*Laboratory for Internet Software Technologies
Institute of Software, Chinese Academy of Sciences*

Agenda

- Introduction
- The RVSim Approach
- Case Study
- Conclusions and Future Work

Agenda

- *Introduction*
- The RVSim Approach
- Case Study
- Conclusions and Future Work

Introduction

- Requirements Volatility
 - One of the most frequent and severe risks
 - Great Negative impacts on software projects
 - Inevitable in software projects
 - Need to be managed effectively
- Current researches
 - Impact of requirements volatility on development productivity, project effort, cost, schedule, product quality, customer satisfaction, software release, change effort and so on

Introduction

- Our focus
 - Impact of requirements volatility on software project plans
- Project plans
 - Very important for software project management
 - Usually made based on software requirements
 - Tasks have strong connections with requirements

Introduction

- When requirements change
 - Influence tasks related to the requirements
 - Influence other related requirements
- Our solution: RVSim (Requirements Volatility Simulation)
 - Predict the impact on project plans
 - Present the plans after adjusting
 - Provide information for decision support

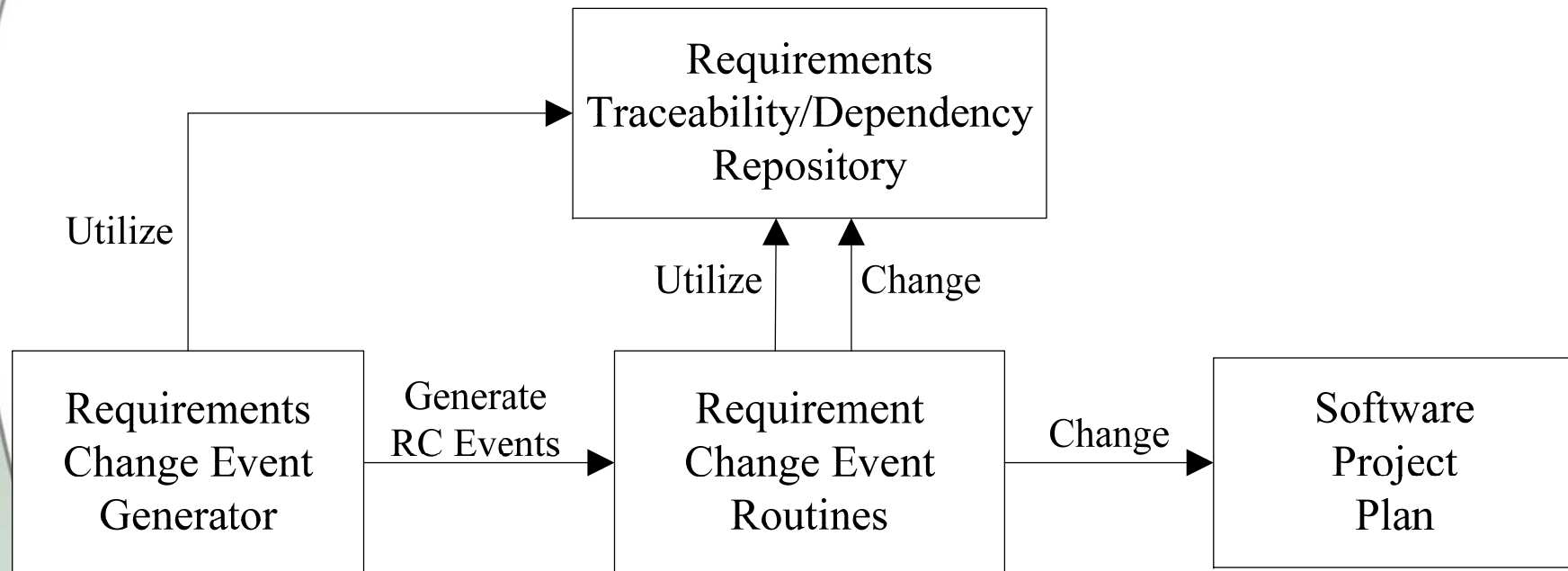
Agenda

- Introduction
- *The RVSim Approach*
- Case Study
- Conclusions and Future Work

The RVSim Approach

- Features
 - Discrete-Event Simulation
 - Requirements changes as events
 - Using requirements traceability/dependency information
 - Traceability: Information between requirements and work products/tasks
 - Dependency: Information between requirements themselves
 - Two requirements volatility input types
 - Definite requirement changes
 - Requirement volatility trends

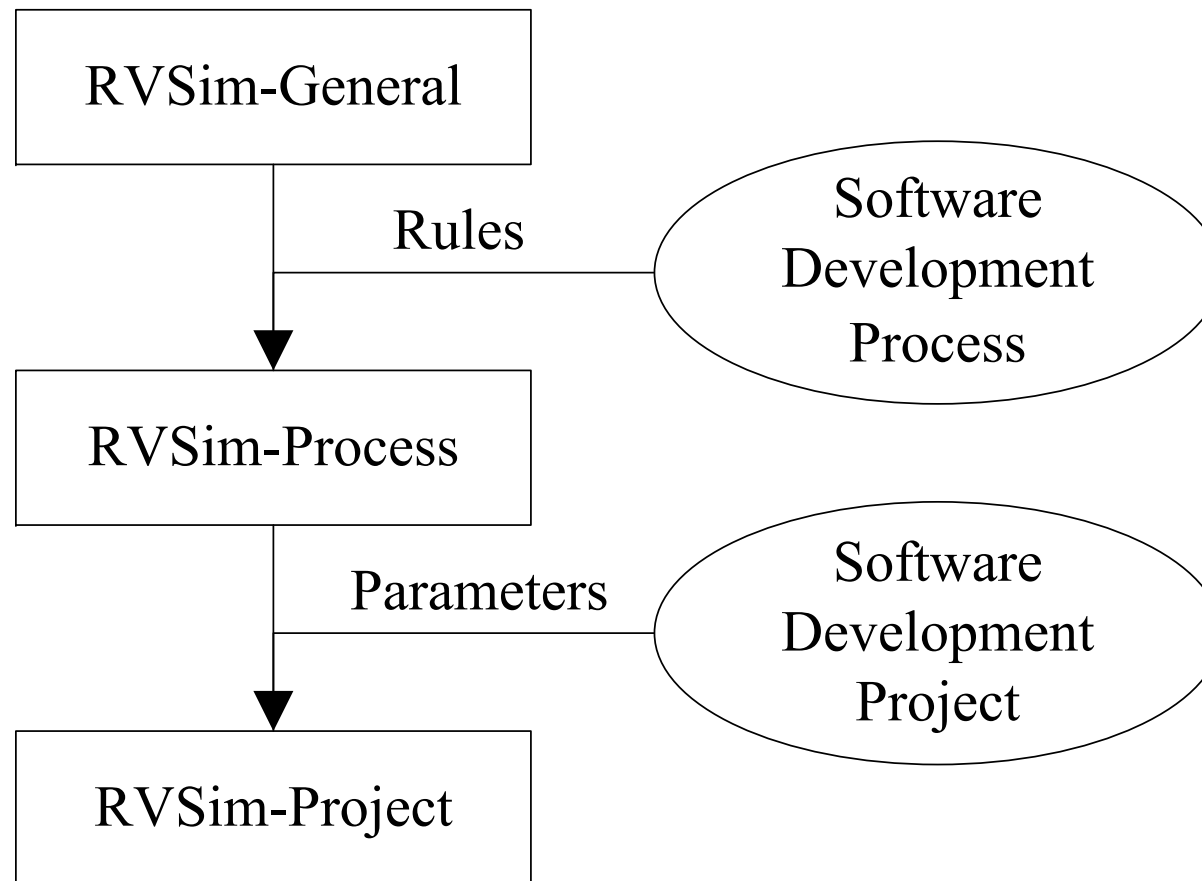
RVSim Structure



RVSim Structure

- Four components:
 - Requirements Traceability/Dependency Repository: store requirements traceability and dependency information
 - Requirements Change Event Generator: generate requirements change events
 - Requirements Change Event Routines: handle the requirements change events
 - Software Project Plan: display the adjusted project plan

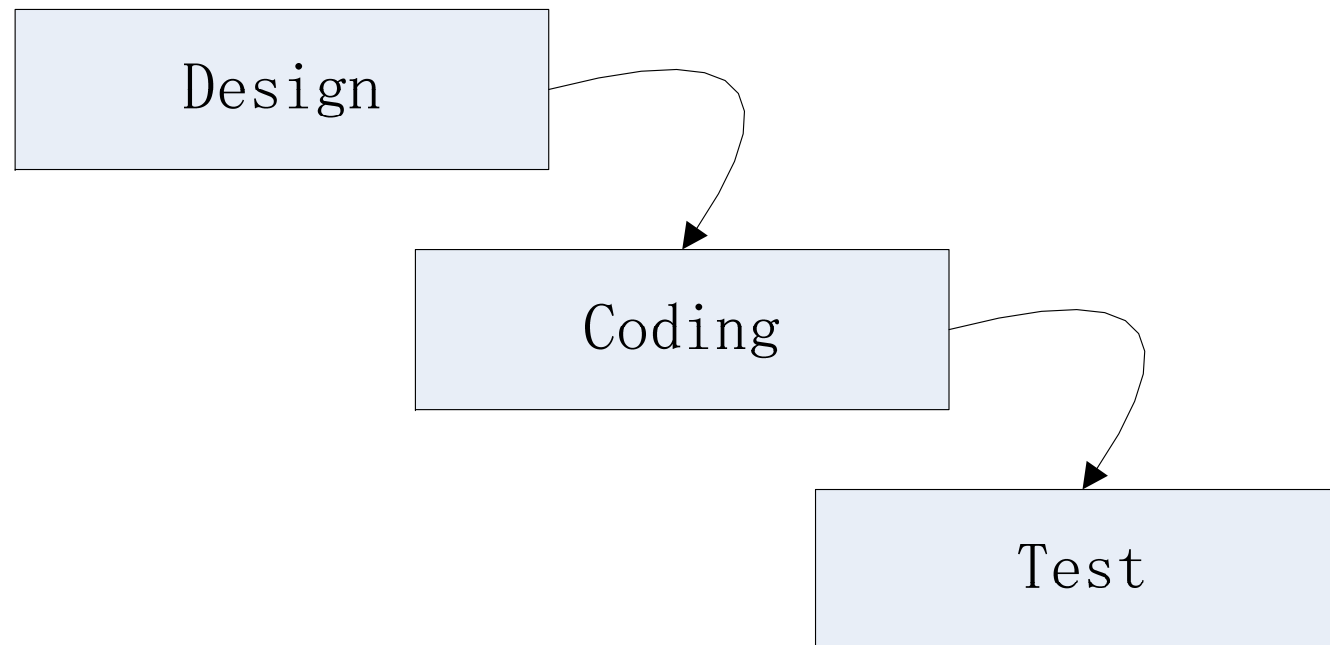
RVSim Abstraction Levels



RVSim Abstraction Levels

- Three abstraction levels
 - RVSim-General: framework of the approach
 - General expressions of the four components
 - Customizable points
 - Guidelines for user customization
 - RVSim-Process: simulation model for one concrete software process
 - Rules of the software process
 - RVSim-Project: simulation model for one concrete software project
 - Parameters of the software project

Example: A Simplified Software Process



The Requirements Traceability/Dependency Repository

- $RTDR = \{RTDI_1, RTDI_2, \dots, RTDI_N\}$
- $RTDI = (RequirementID, Volatility, WorkProductSet, DependencySet)$
 - $WorkProductSet = \{WorkProduct_1, WorkProduct_2, \dots, WorkProduct_M\}$
 - $WorkProduct = (Type, Size, SizeUnit)$
 - $DependencySet = \{Dependency_1, Dependency_2, \dots, Dependency_S\}$
 - $Dependency = (RequirementID, DependencyStrength)$

The Requirements Traceability/Dependency Repository

- **Three Customizable Points**
 - *VolatilitySet*: contains all possible values for *Volatility*
 - *DependencyStrengthSet*: contains all possible values for *DependencyStrength*
 - *WorkProductSet*

The Requirements Traceability/Dependency Repository

- Customize the example process
 - $VolatilitySet = \{high, low\}$
 - $DependencyStrengthSet = \{strong, weak\}$
 - $WorkProductSet = \{DesignDocs, Codes, TestCases\}$
 - $DesignDocs = (DesignDocument, ds, page)$
 - $Codes = (Code, cs, LOC)$
 - $TestCases = (TestCase, ts, test\ case)$

The Requirements Change Event Generator

- Requirement change event
 - $RCEvent = (RCType, RCTime, RTDI, ModificationLevel)$
 - $RCType$: RA (Requirements Addition), RM (Requirements Modification) or RD (Requirements Deletion)
- Two modes for generating events
 - Definite events inputted by users
 - Supposed events generated according to requirements volatility trends

The Requirements Change Event Generator

- **Two Customizable Points**
 - *ModificationLevelSet*: contains all possible values for *ModificationLevel*
 - **Mode for generating events**
 - Rules for the requirements volatility trends mode

The Requirements Change Event Generator

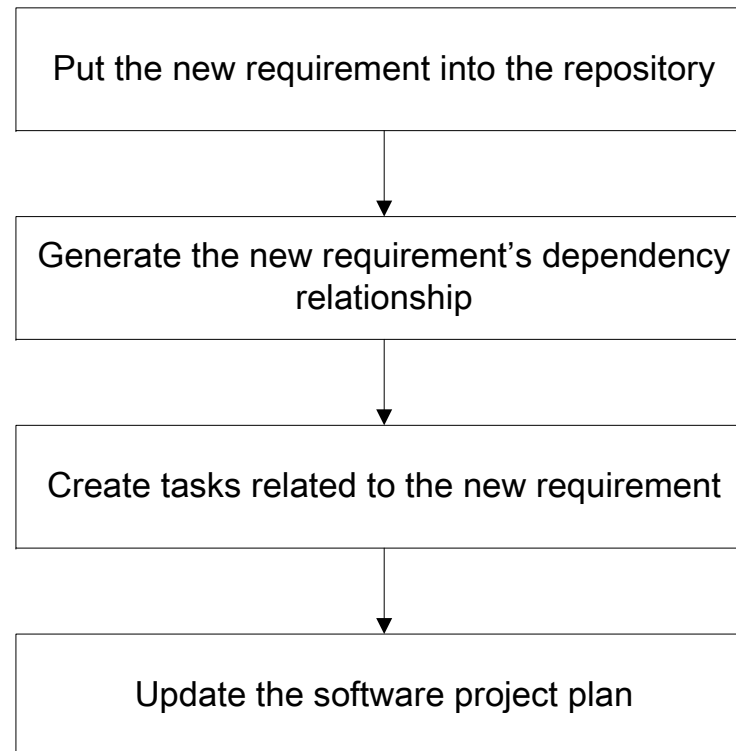
- Customize the example process
 - $ModificationLevelSet = \{delete, major, moderate, minor\}$
 - The second mode
 - Rules for generating events
 - Define p ($p > 0$): intervals between requirements changes
 - Define q ($q > 0$): period requirements changes happen
 - Define ap, mp, dp ($ap \geq 0, mp \geq 0, dp \geq 0, ap + mp + dp = 100\%$): percents of three kinds of events
 - For RM , define map, mop, mip ($map \geq 0, mop \geq 0, mip \geq 0, map + mop + mip = 100\%$): percents of *ModificationLevel* value.

The Requirements Change Event Generator

- Rule for generating events (Cont.)
 - For *RA*, generate a new *RTDI*
 - *RequirementID*: generated automatically
 - *Volatility*: chosen randomly
 - *DependencySet*: empty
 - *Size* for *DesignDocument*, define *dll*, *dul*: lower and upper limits
 - *Size* for *Code*, define *cll*, *cul*: lower and upper limits
 - *Size* for *TestCase*, define *tll*, *tul*: lower and upper limits
 - *Size* is generated randomly between lower and upper limits
 - For *RM* and *RD*, choose *volatility* value randomly

The Requirements Change Event Routines

- Requirements Addition Event Routine: 4 steps



The Requirements Change Event Routines

- Three Customizable Points
 - The rule for generating dependency relationship
 - The rule for creating tasks
 - The rule for adjusting the software project plan

The Requirements Change Event Routines

- Customize the example process
 - The rule for generating dependency relationship
 - Define N : total number of requirements
 - Define NA : number of requirements the new one depends
 - Define $dper = (NA/N) * 100\%$
 - Define $dperll$, $dperul$: the lower and upper limits of $dper$
 - $dper$ is generated randomly between $dperll$ and $dperul$.
 - *DependencyStrength*: chosen randomly

The Requirements Change Event Routines

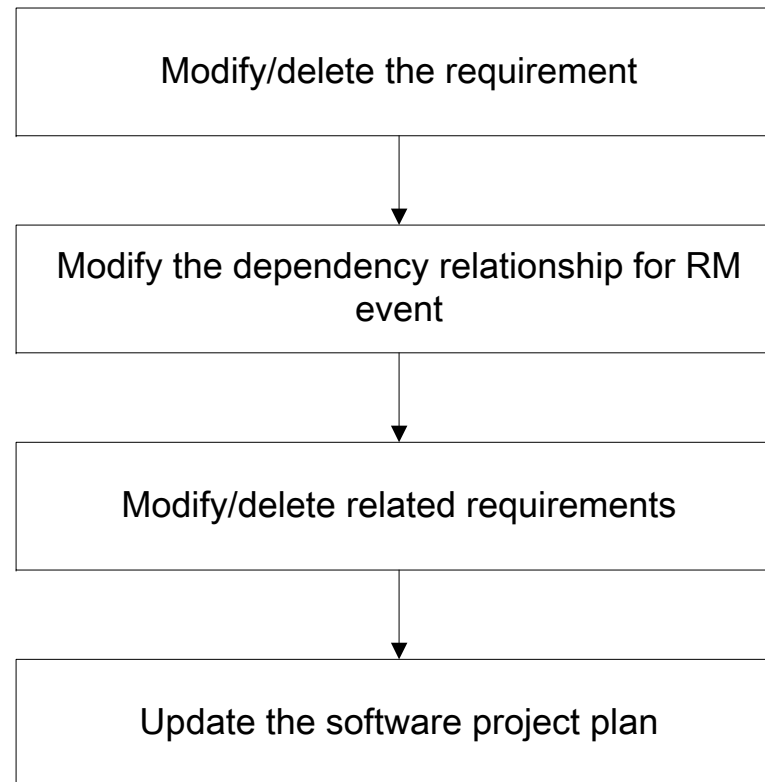
- Customize the example process
 - The rule for creating tasks
 - Define *dpro page/hour*: design task productivity
 - Define *cpro LOC/hour*: coding task productivity
 - Define *tpro test case/hour*: test task productivity
 - Create three new tasks: *design task*, *coding task*, *test task*. The time for these tasks are calculated by productivity and work product sizes.

The Requirements Change Event Routines

- Customize the example process
 - The rule for adjusting the software project plan
 - Different phases can not overlap.
 - Design tasks have precedence relationship the same as the dependency of requirements related to them.
 - There is no idle time between tasks.

The Requirements Change Event Routines

- Requirements Modification/Deletion Event Routine: 4 steps



The Requirements Change Event Routines

- Three Customizable Points
 - The rule for modifying requirements
 - The rule for modifying dependency relationship
 - The rule for deciding *ModificationLevel* for related requirements

The Requirements Change Event Routines

- Customize the example process
 - The rule for modifying requirements
 - *major, moderate, minor*: correspond to numeric values between 0 and 1, $major \geq moderate \geq minor$.
 - Define *smp*: size modified percent
 - If $ModificationLevel = major$, $moderate < |smp| \leq major$
 - If $ModificationLevel = moderate$, $minor < |smp| \leq moderate$
 - If $ModificationLevel = major$, $0 < |smp| \leq minor$
 - *smp* is generated randomly in its range

The Requirements Change Event Routines

- Customize the example process
 - The rule for modifying dependency relationship
 - Define $dpermp$: modified percent of $dper$
 - $0 \leq |dpermp| \leq ModificationLevel$
 - $dpermp$ is generated randomly in its range.

The Requirements Change Event Routines

- Customize the example process
 - The rule for deciding *ModificationLevel* for related requirements

R_{MD} 's <i>ModificationLevel</i>	<i>DependencyStrength</i>	R_d 's <i>ModificationLevel</i>
<i>delete</i>	<i>strong</i>	<i>delete</i>
<i>delete</i>	<i>weak</i>	<i>major</i>
<i>major</i>	<i>strong</i>	<i>major</i>
<i>major</i>	<i>weak</i>	<i>moderate</i>
<i>moderate</i>	<i>strong</i>	<i>moderate</i>
<i>moderate</i>	<i>weak</i>	<i>minor</i>
<i>minor</i>	<i>strong</i>	<i>minor</i>
<i>minor</i>	<i>weak</i>	<i>none</i>

- ◆ R_{MD} : the modified or deleted requirement
- ◆ R_d : requirements depending on R_{MD}

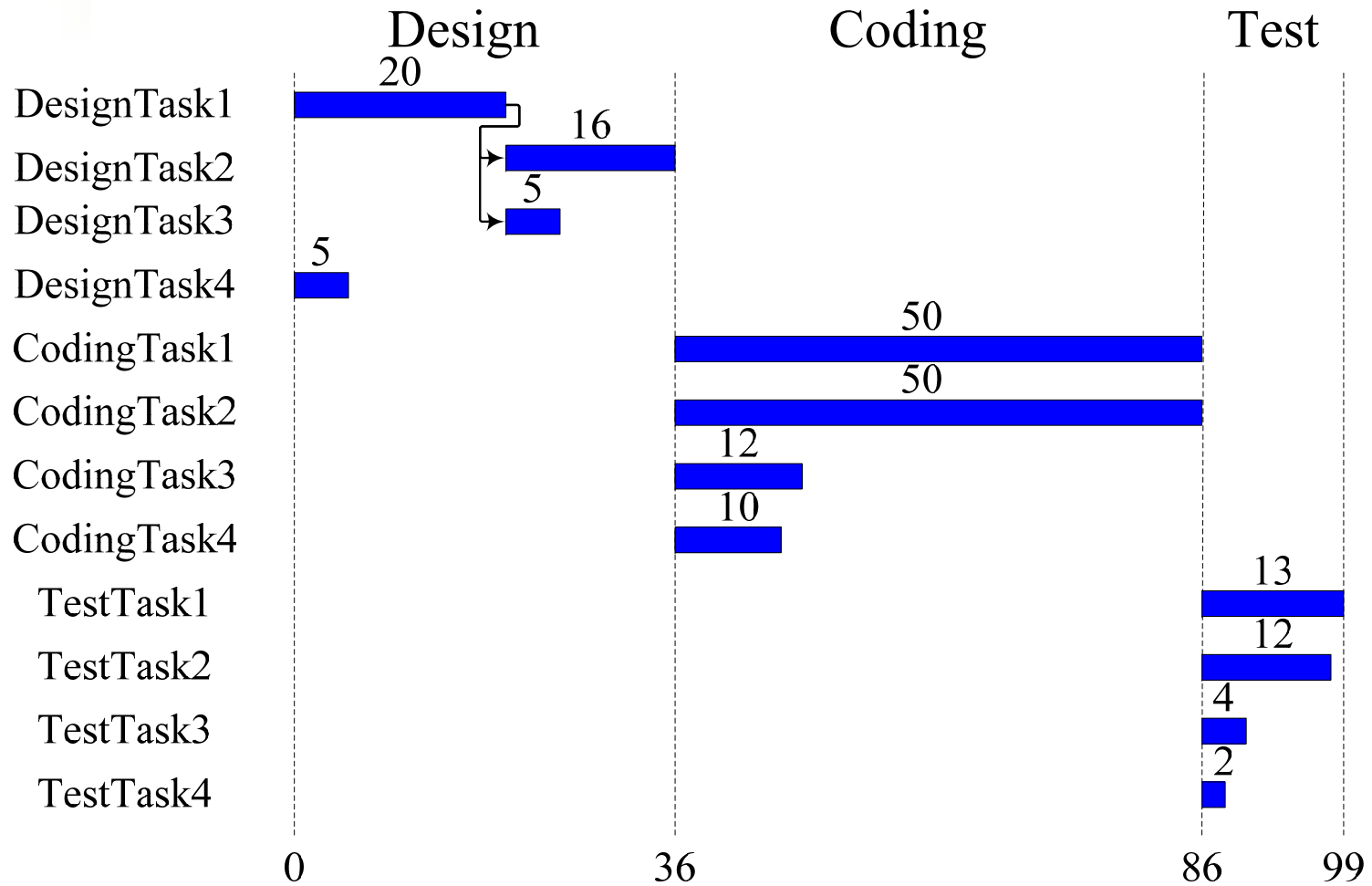
Agenda

- Introduction
- The RVSim Approach
- *Case Study*
- Conclusions and Future Work

Example Project

- Background
 - Requirements: 4
 - R2 and R3 depends on R1
 - Deadline: 120 working hours
- Assumptions
 - Enough human resources
 - No other risk factors
- Question
 - Can the project be finished before deadline?

Original Project Plan



Example Project Parameters

- $p=16, q=40$
- $ap=25\%, dp=25\%, mp=50\%$
- $dul=8, dll=2, cul=4000, cll=1000, tul=30, tll=10$
- $map=10\%, mop=20\%, mip=70\%$
- $major=0.8, moderate=0.5, minor=0.25$
- $dperul=30\%, dperll=0$
- $dpro=0.4, cpro=50, tpro=2$

Example Project Requirements

- $RTDR = \{RTDI1, RTDI2, RTDI3, RTDI4\}$
 - $RTDI1 = (1, low, \{(DesignDocument, 6, page), (Code, 2000, LOC), (TestCase, 25, test\ case)\}, \{(2, strong), (3, strong)\})$
 - $RTDI2 = (2, low, \{(DesignDocument, 8, page), (Code, 3000, LOC), (TestCase, 30, test\ case)\}, \{\})$
 - $RTDI3 = (3, high, \{(DesignDocument, 2, page), (Code, 600, LOC), (TestCase, 6, test\ case)\}, \{\})$
 - $RTDI4 = (4, low, \{(DesignDocument, 1.5, page), (Code, 500, LOC), (TestCase, 5, test\ case)\}, \{\})$

Demo: one RA event

- **Event**

- $RCEvent = (RA, 16, RTDI5, null)$
- $RTDI5 = (5, high, (DesignDocument, 6.96, page), (Code, 2032, LOC), (TestCase, 26, test case), \{\})$

- **Handling**

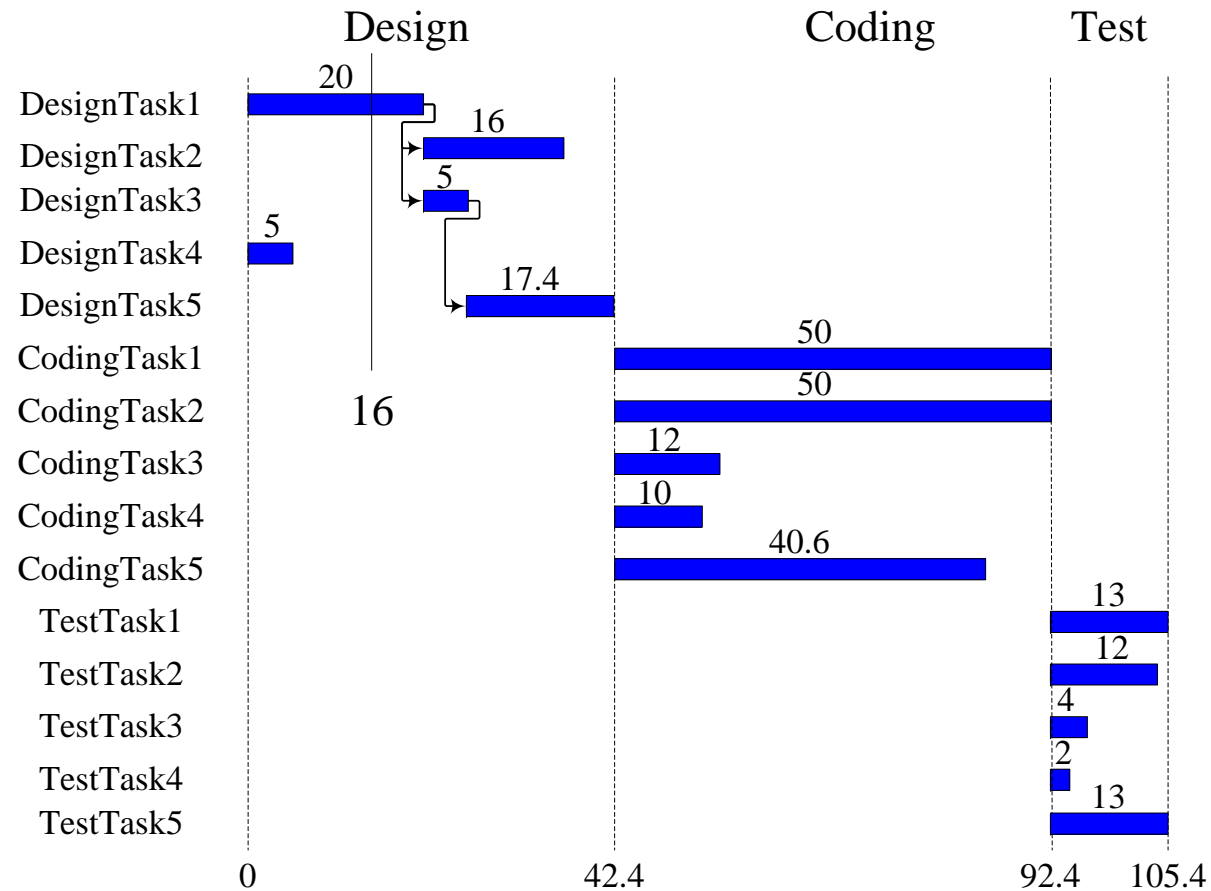
- Generate randomly $dper = 21.04\%$
- $R5$ depends on one requirement, randomly choose $R3$
- Create new tasks for $R5$
 - $DesignTask5: 6.96/0.4 = 17.4$ hours
 - $CodingTask5: 2032/50 = 40.6$ hours
 - $TestTask5: 26/2 = 13$ hours

Demo: one RA event (Cont.)

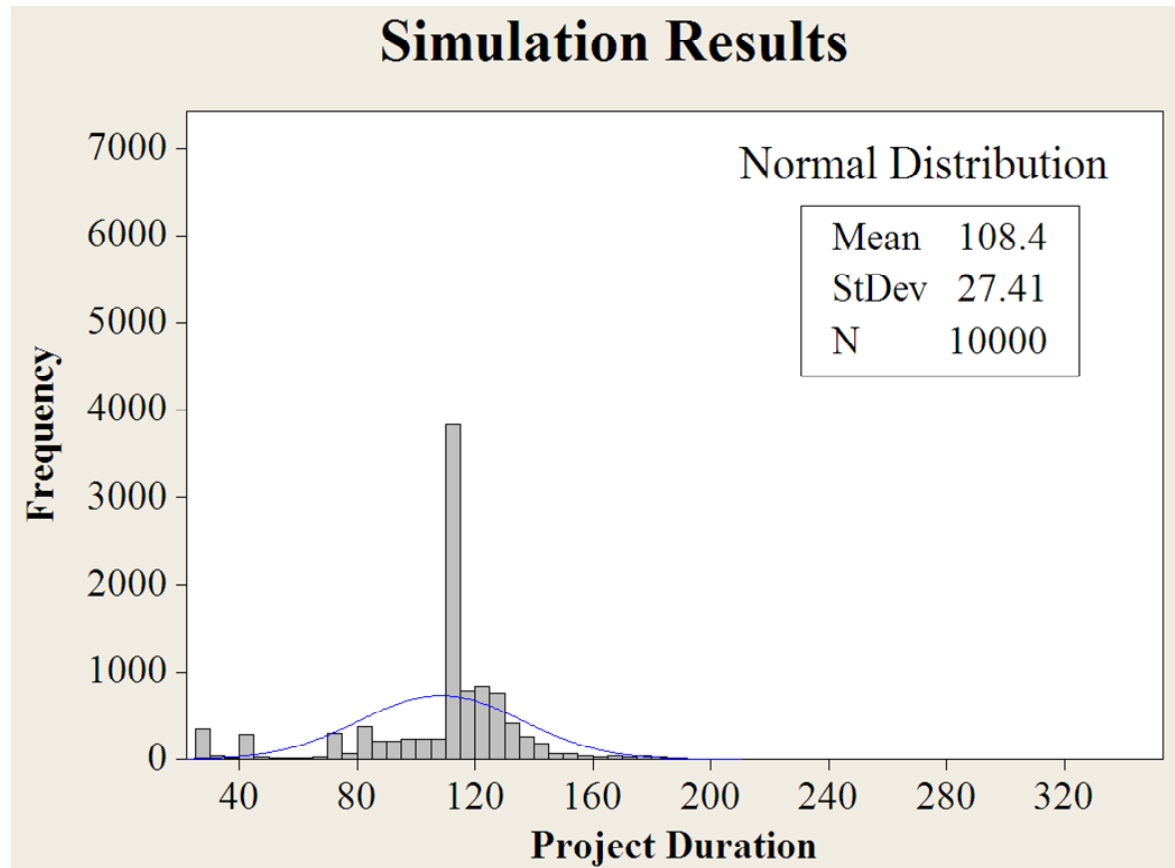
- After the event
 - $RTDR = \{RTDI1, RTDI2, RTDI3, RTDI4, RTDI5\}$
 - $RTDI3 = (3, high, \{(DesignDocument, 2, page), (Code, 600, LOC), (TestCase, 6, test\ case)\}, \{(5, weak)\})$

Demo: one RA event (Cont.)

- Project plan after the event

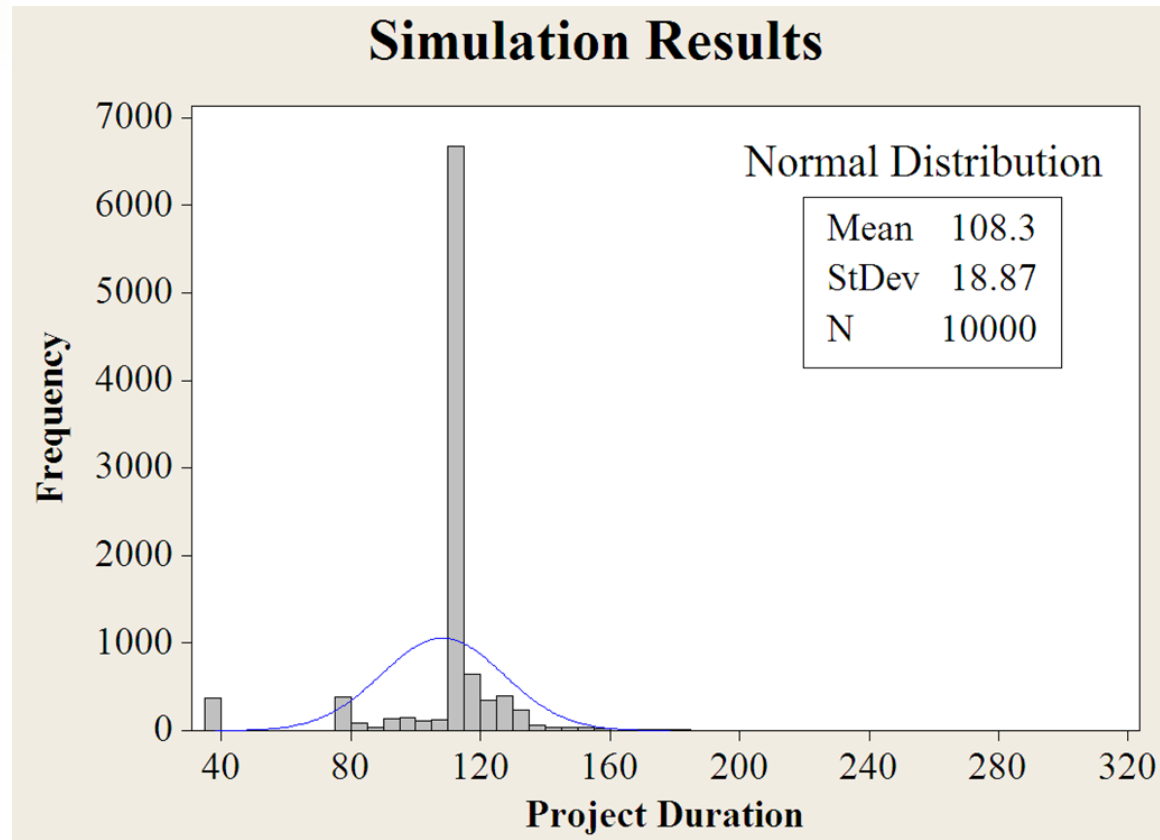


Simulation Results



10000 times, the probability the project will be finished before deadline is 66.3%

Simulation Results



change $p=24$. 10000 times, the probability the project will be finished before deadline is 73.2%

Agenda

- Introduction
- The RVSim Approach
- Case Study
- *Conclusions and Future Work*

Conclusion and Future Work

- RVSIm approach
 - Predict the impact of requirements volatility on project plans
 - Utilize requirements traceability and dependency information
 - Display the impacted software project plan
 - Support user customization and reuse
- Future Work
 - Some assumptions are not always realistic
 - Apply to more software projects, collect feedback and improve the approach

Thank You!