

Achieving On-Time Delivery: A Two-Stage Probabilistic Scheduling Strategy for Software Projects

Xiao Liu¹, Yun Yang¹, Jinjun Chen¹, Qing Wang², and Mingshu Li²

¹Centre for Complex Software Systems and Services

Swinburne University of Technology

²Laboratory for Internet Software Technologies

Chinese Academy of Sciences

Presented by Prof. Yun Yang
yyang@swin.edu.au



Content



■ Introduction

- Uncertainty of Software Processes
- On-Time Delivery

■ A Two-Stage Probabilistic Scheduling Strategy

- Motivation and Strategy Overview
- Stage 1: Setting deadlines
- Stage 2: GA based project scheduling

■ Evaluation

■ Conclusion



Introduction

- Uncertainty of software processes
 - Activities completed under or over schedule
 - Resources become unavailable
 - Change of schedules: brought forward or postponed
 - Change of requirements: added or removed
 - Many more...
- Schedule (and cost) estimation under uncertainty
- Project scheduling under uncertainty

Introduction



■ On-time delivery

- On-time delivery of core capabilities is increasingly becoming the main focus of software processes in the dynamic business world nowadays

■ Customers

- require fast development of small scale software components within hard deadlines to meet their dynamic and urgent business needs

■ Small and medium sized software development organisations

- main business targets: frequent short-term contracts from a relatively fixed group of customers with high loyalty in the markets
- main challenges: on-time delivery of individual software products given multiple software projects concurrently running in the organisation



Where Are We

■ Introduction

- Uncertainty of Software Processes
- On-Time Delivery

■ A Two-Stage Probabilistic Scheduling Strategy

- Motivation and Strategy Overview
- Stage 1: Setting deadlines
- Stage 2: GA based project scheduling

■ Evaluation

■ Conclusion

Motivation



- Practical data show that about one-third of the projects exceed their estimated schedules by 25%*
 - A project schedule is not well balanced between the software process performance baseline and customer needs
 - far beyond the software process performance baseline
 - emphasises the role of project managers to estimate schedules but neglect the role of customers
 - An individual baseline schedule does not consider the situation of multiple software processes
 - the competition of employees among multiple software processes
 - individual baseline schedules cannot guarantee on-time delivery without considering multiple software processes

Strategy Overview



Two-Stage Probabilistic Project Scheduling Strategy

Overview	Input: Process models, Historic project information Method: Two-stage probabilistic project scheduling strategy Output: Project schedules for on-time delivery
Stage1: Pre-scheduling	Step1.1: Modelling software processes with Stochastic Petri Nets
	Step1.2: Setting deadlines for individual software processes with a probability based temporal consistency model supported negotiation between customers and project managers
Stage2: Scheduling	Step2.1: Minimising the overall completion time of multiple software processes with GA based searching
	Step2.2: Searching for the optimal or near optimal solution which meets all deadlines

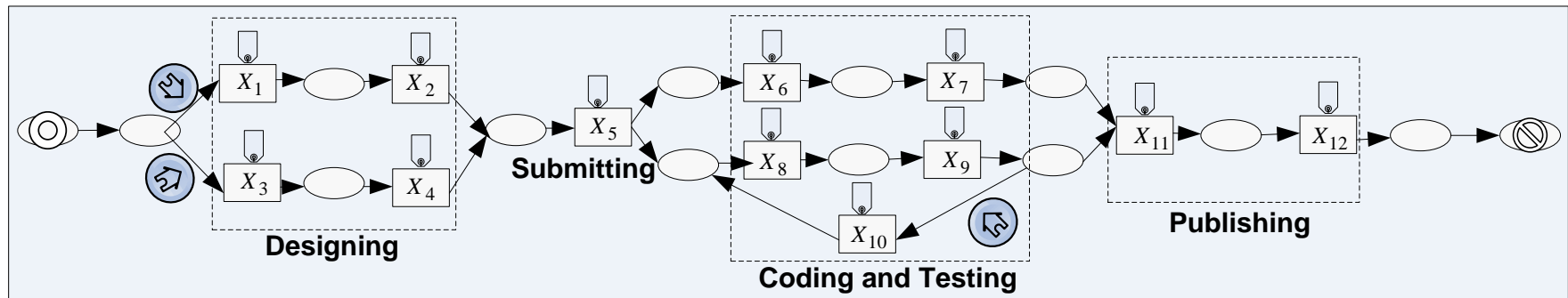
Stage 1: Setting Deadlines



- Step 1.1: Modelling software processes with Stochastic Petri Nets



Graphic Notations



An Example Process Model



Stage 1: Setting Deadlines

- Step 1.2: Setting deadlines for individual software processes based on win-win negotiation between customer and project managers
- Probability based temporal consistency model

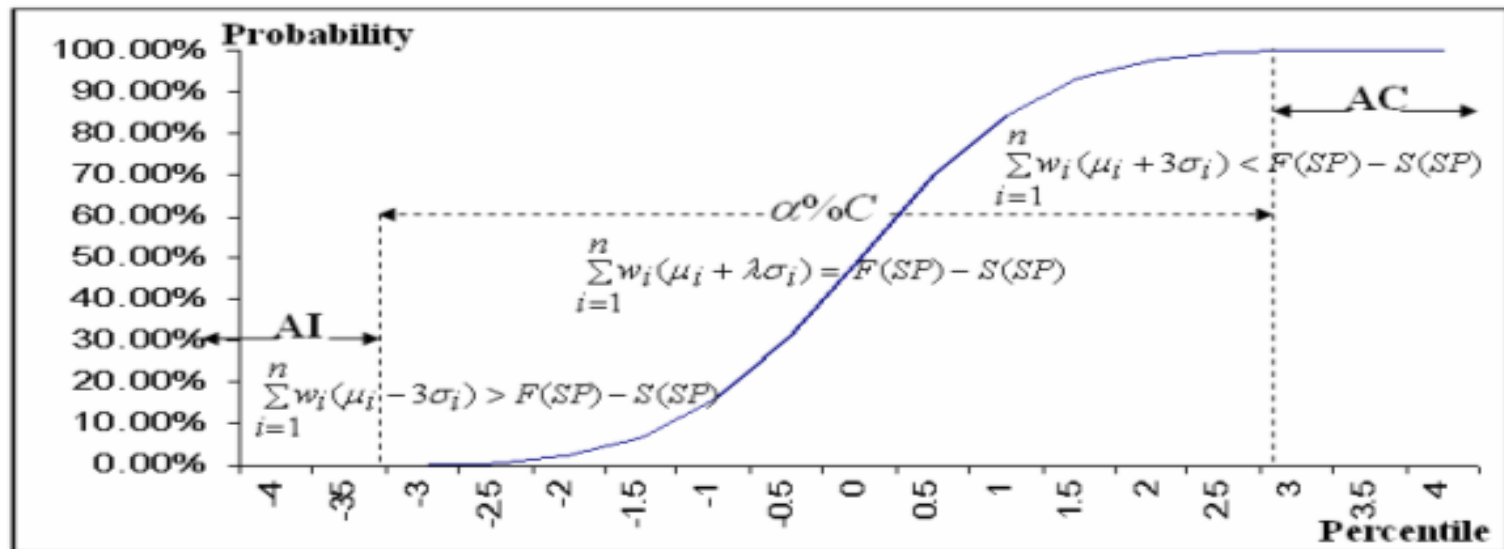


Figure. Probability Based Temporal Consistency

Probabilistic Setting Strategy*

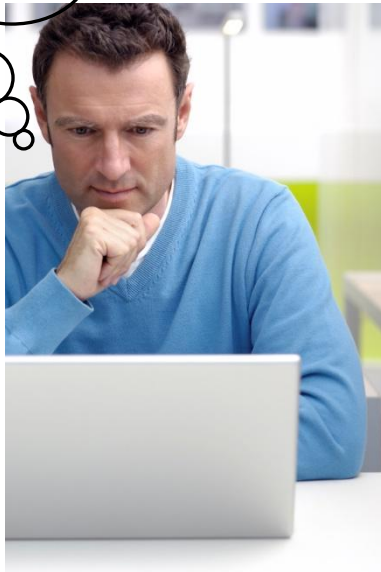


Probabilistic strategy for setting temporal constraints	
Overview	<p>Input: Process model and system logs for scientific workflow SW</p> <p>Method: Probabilistic setting strategy</p> <p>Output: Coarse-grained upper bound constraints and fine-grained upper bound constraints</p>
<p>Step1: Calculating weighted joint distribution</p>	Obtain the statistic information (activity duration distribution $N(\mu_i, \sigma_i^2)$ and weight w_i) from workflow system logs; Calculate the weighted joint distribution of the workflow by the composition of basic building blocks.
<p>Step2: Setting coarse-grained constraints</p>	Set coarse-grained temporal constraints through the negotiation process with the weighted joint distribution $N(\mu_{sw}, \sigma_{sw}^2)$ and the probability based temporal consistency.
<p>Step3: Setting fine-grained constraints</p>	Set fine-grained temporal constraints as $\mu_i + \lambda\sigma_i$ for activity a_i with $N(\mu_i, \sigma_i^2)$, where λ is the $\alpha\%$ percentile for the achieved coarse-grained temporal constraints.

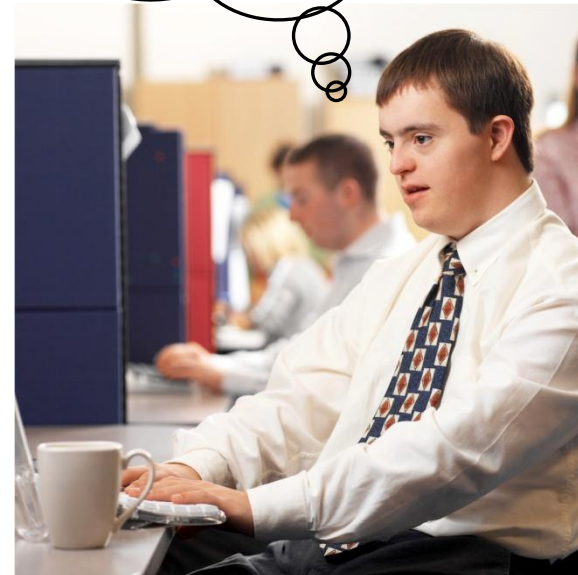


Example: Setting Deadlines

I Want the process be completed in 48 hours



Let me check the probability

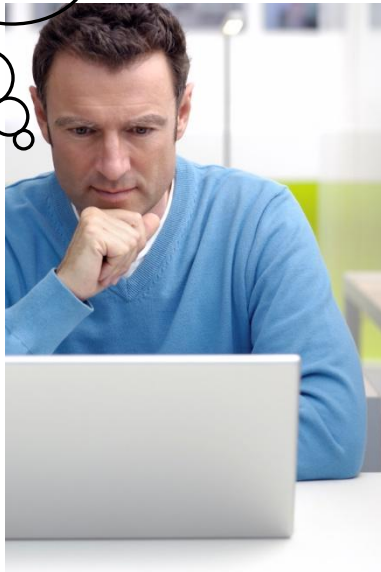


The negotiation process

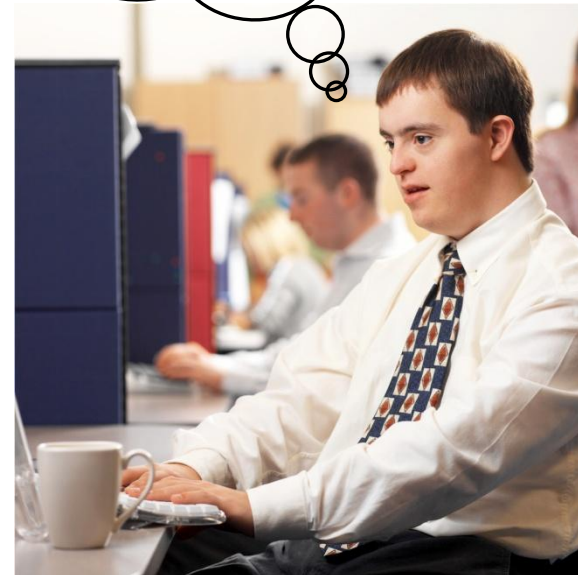


Example: Setting Deadlines

That's not good, how about 52 hours



Sir, its 70%, do you agree?



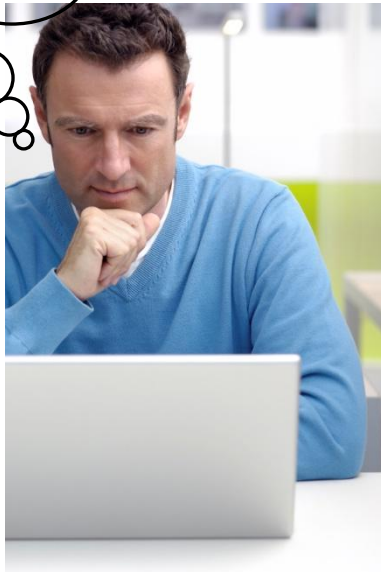
Adjust the constraint



Example: Setting Deadlines

Err... how long will it take if I want to have

90%



Then, it increases to 85%

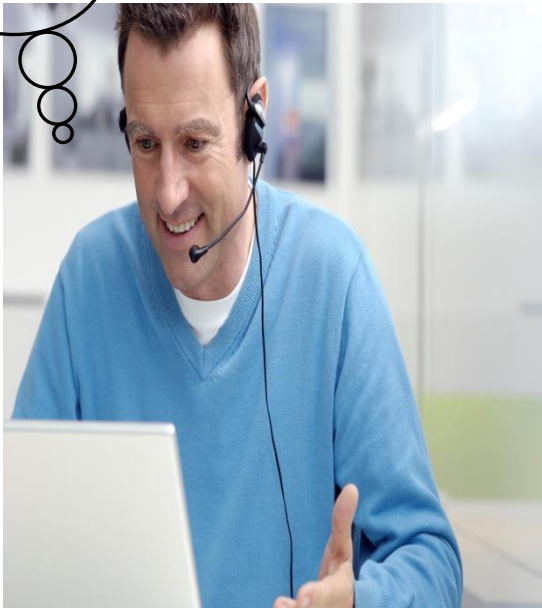


Adjust the probability

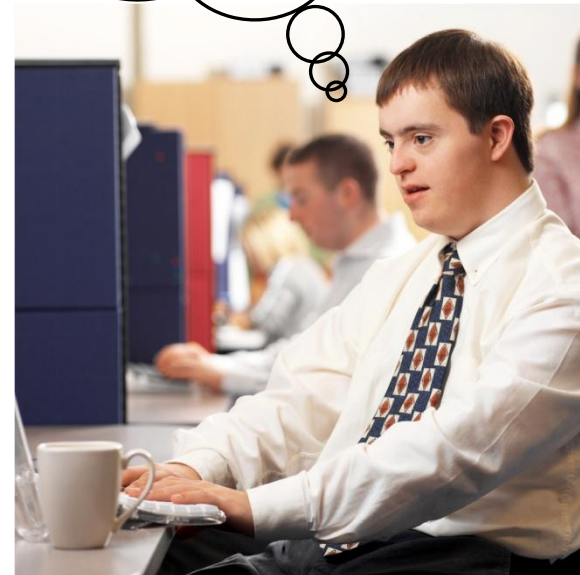


Example: Setting Deadlines

Ok, that's the deal! Let's do it!



It will take us 54 hours



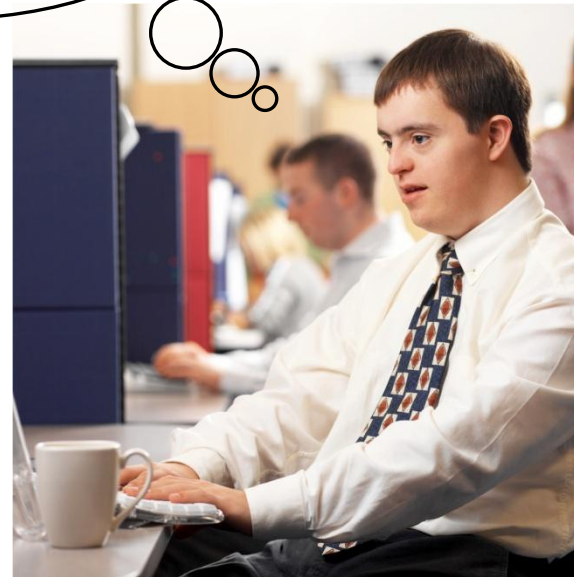
Negotiation result



Example: Setting Deadlines

Ok! But, sir, I need to remind you that this is only a guarantee from statistic sense. If we cannot make it, please blame the guy who comes up with the strategy!

Sorry, statistically, no predictions can be 100% sure!



Stage 2: Project Scheduling



■ Problems

- I : How to achieve on-time delivery for individual software processes while minimising the overall completion time for multiple software processes at the meantime.
 - II : How to realise practical restrictions, e.g. implementing heuristic rules in GA (genetic algorithm).
- In real-world software projects, there are many restrictions which affect the task-to-employee assignment.
- Heuristic rules such as resource continuity (assigning a group of highly related tasks, e.g. requirement acquisition and function design, to a fixed resource so as to reduce the overhead of task transfer and promote the efficiency), adjustment of workload and adjustment of overstaffed projects, and many more *

*Chang, C.K., Jiang, H., Di, Y.: Time-line Based Model for Software Project Scheduling, Information and Software Technology (2008) ¹⁶
X. Liu, Y. Yang, J. Chen, Q. Wang, and M. Li, Achieving On-Time Delivery(ICSP09), Vancouver, Canada, 16-17 May 2009

GA based Two-Phase Project Scheduling



- To tackle problem I
- The first phase is to optimise the overall completion time of multiple software processes. Based on genetic operations, e.g. selection, crossover and mutation, the searching space is expanded and solutions with higher fitness values are found.
 - The fitness value is defined according to the overall completion time of all the software processes. The smaller the overall completion time, the higher the fitness value is.
- The second phase is to search for the best solution from the whole solution set composed of the best child in each generation produced in the first searching phase. The best solution found should meet the deadlines of individual software processes while having the minimum overall completion time.

Package based initialisation approach



- To tackle problem II
- For a specific package, a set of employees are formed first by checking required abilities. Afterwards, a further checking process is applied to select valid employees based on enforced heuristics. Finally, one of the valid employees is randomly assigned to this package.

Number of packages = k
 Number of tasks = n
 Number of resources = p

	Package 1			Package 2		Package k-1			Package k			
Task Sched _i	1	2	3	4	5	...	m-2	m-1	m	m+1	m+2	...	n
Resource alloc _i	R_2	R_2	R_2	R_3	R_3	...	R_1	R_1	R_1	R_p	R_p	...	R_p

High Level Pseudo Code



Strategy: GA based project scheduling

Input: Software process models
Employee models

Output: Generated Scheduling Plans

//two dimensional encoding

1) ENCODING(L);

// package based generation of initial populaton

2) INITIALISATION($Population$)

// Optimising overall completion time

3) **While** (maximum generation is not reached)

{

//selecting *Parents* from *Population*

4) SELECTION($Population$);

5) CRSSSOVER($Parents$)→ $Children$;

//change resource for a random task

6) MUTATION($Children$);

//check with software process models and heuristic re

7) VALIDATION($Children$);

//store the best child into a solution set; replace worst

//with the best child

8) STORE($BestChild$, $SolutionSet$);

9) REPLACE($WorstChild$, $BestChild$);

}

10) **Return** $SolutionSet$ and $BestChild$ in $Popu$

// Searching for the BestSolution from solution set

11) $BestSolution = BestChild$;

12) **While** (not end of the $SolutionSet$)

{

//Compare each solution and set $BestSolution$ as the

//individual deadlines and has smaller overall comple

13) $BestSolution = COMPARE(BestSolution,$

}

14) **Return** the $BestSolution$;

// decoding and update

15) DECODING($BestSolution$);

16) UPDATE(L).

Function: INITIALISATION

Input: Software process models $SP\{SP_i(1 \leq i \leq k)\}$

Employee models $EP\{EP_i(1 \leq i \leq p)\}$

Output: $sched_i, alloc_i$: chromosome

1) **for** $i = 1$ **to** size // generate a fixed size of solutions

// generate a single solution

2) **for each** Software Process SP

// arrange tasks to form packages

// function size() returns the number of tasks in each SP

3) $sched_i \left(\sum_{m=1}^{j-1} size(SP_m) + 1 : \sum_{m=1}^j size(SP_m) \right)$
 $= task \left(\sum_{m=1}^{j-1} size(SP_m) + 1 : task \left(\sum_{m=1}^j size(SP_m) \right) \right);$

// allocate employees to each package

4) **for each** Package P in each Software Process SP

// check package requirements with employee models and

// return the matched employee set $EP'\{\}$

5) $EP'\{\} = Check(Package, EP\{\})$

// apply heuristic rules $R\{\}$ and return the valid employee set $EP''\{\}$

6) $EP''\{\} = Check(R\{\}, EP'\{\})$

// random allocate valid employees to the package

7) $alloc_i \left(\sum_{m=1}^{j-1} size(SP_m) + 1 : \sum_{m=1}^j size(SP_m) \right)$
 $= random(EP''\{\});$

8) **end**

9) **end**

10) **end**



Where Are We

■ Introduction

- Uncertainty of Software Processes
- On-Time Delivery

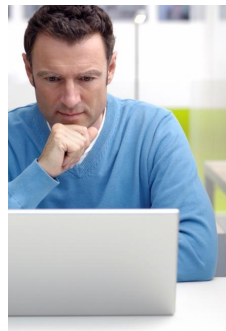
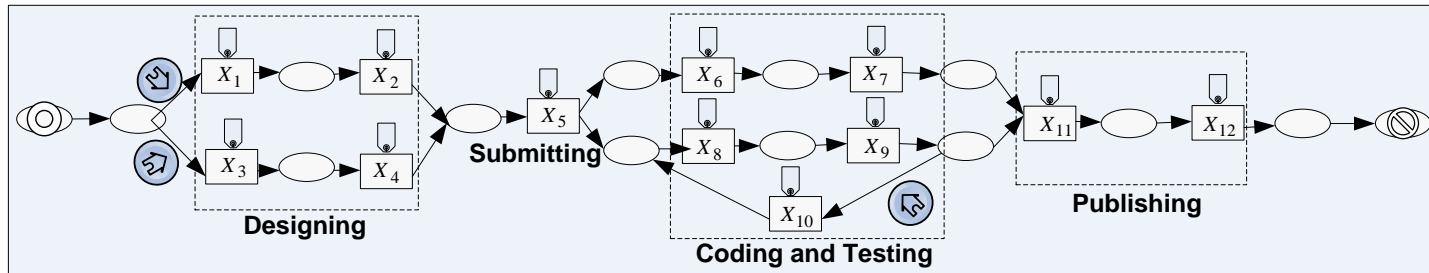
■ A Two-Stage Probabilistic Scheduling Strategy

- Motivation and Strategy Overview
- Stage 1: Setting deadlines
- Stage 2: GA based project scheduling

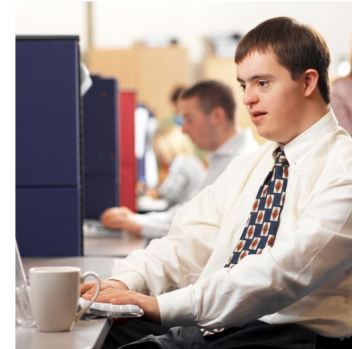
■ Evaluation

■ Conclusion

Setting Deadlines



Negotiation



Example Software Process				Pseudo-Scheduling	
Task	Mean	Variance	Weight	Process Segments	Weighted Joint Distribution
Task X_1	105	225	0.67	Designing: Choice process. The probability for the upper path is 66.7%, the lower path is 33.3%.	$Mean = 0.67 * (105 + 223) + 0.33 * (256 + 358) = 422$ $Variance = 0.67^2(225 + 289) + 0.33^2(529 + 400) = 331$
Task X_2	223	289	0.67		
Task X_3	256	529	0.33		
Task X_4	358	400	0.33		
Task X_5	558	784	1	Submitting: Sequence process	$Mean = 558$; $Variance = 784$
Task X_6	650	1089	0	Coding and Testing: Parallelism and iteration process. The statistic iteration times of the upper path are 5 and the lower path are 4.	$Mean = 5 * (125 + 285) + 4 * 594 = 4426$ $Variance = 5^2 * (64 + 1444) + 4^2 * 484 = 45444$
Task X_7	230	225	0		
Task X_8	125	64	5		
Task X_9	285	1444	5		
Task X_{10}	594	484	4		
Task X_{11}	661	529	1	Publishing: Sequence process	$Mean = 661 + 123 = 784$; $Variance = 529 + 64 = 593$
Task X_{12}	123	64	1		
Overall weight joint distribution				$\mu_{sp} = 422 + 558 + 4426 + 784 = 6190$; $\sigma_{sp} = \sqrt{331 + 784 + 45444 + 593} = 217$ The weighted joint normal distribution for SP $\Rightarrow \mathcal{N}(6190, 217^2)$	
Negotiated Deadline				$f(SP) = 6380$ with 81% consistency and $\lambda = 0.88$	



Simulation Experiment Settings

- Three rounds of experiments with different probability of build-time temporal consistency states COM(1.00), COM(1.15) and COM(1.28) with 84.1%, 87.5%, 90.0% respectively.

Setting for input parameters					Setting for each round of experiment							
Software process models	Stochastic Petri Nets with a random size of (10~20) tasks				Round1	$\lambda_1 = 1.00$ with a probability consistency of 84.1%						
Duration distribution models	Duration distribution of t_j is $N(\mu_j, \sigma_j^2)$ where $\mu = random(30, 3000)$ and $\sigma = 33.3\% * \mu$				Round2	$\lambda_2 = 1.15$ with a probability consistency of 87.5%						
Resource models (Employees or subcontractors)	Execution speed: $R(R_i, ES(R_i))$, resource R_i with the execution speed of $random(1,3)$ where the mean duration of t_j on resource R_i is $\mu_j / ES(R_i)$				Round3	$\lambda_3 = 1.28$ with a probability consistency of 90.0%						
Heuristic rule	Assigning the employee with minimum workload				Setting for T , P and R of each experiment							
GA operations	Maximum Generation	Population Size	Crossover Rate	Mutation Rate	Exp	T	P	R	Exp	T	P	R
	1000	100	0.7	0.1	1	30	2	3	6	140	9	18
					2	50	3	5	7	200	12	22
					3	65	4	8	8	240	15	28
					4	80	5	12	9	280	18	32
					5	110	7	14	10	320	20	36

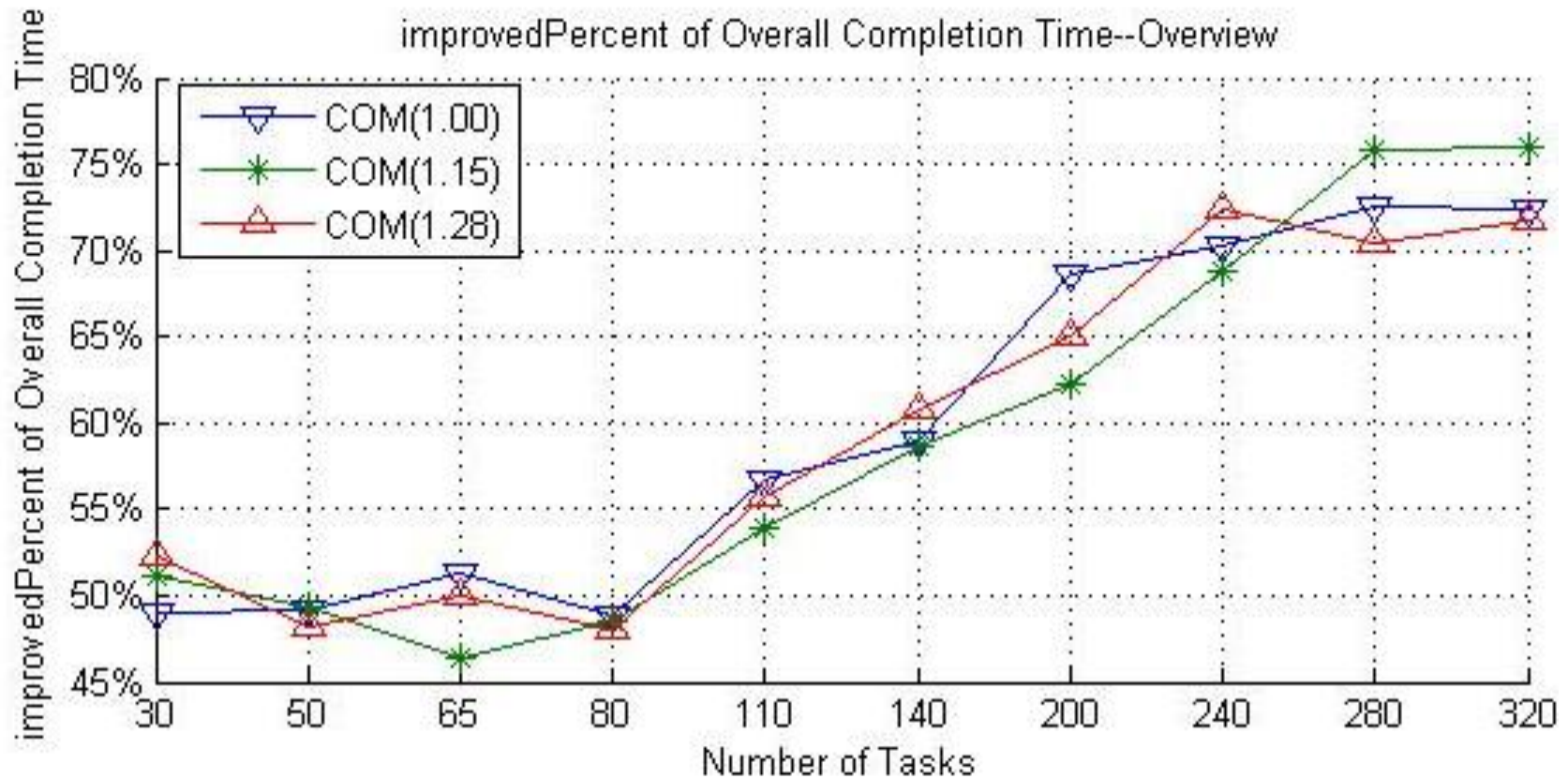
Simulation Results



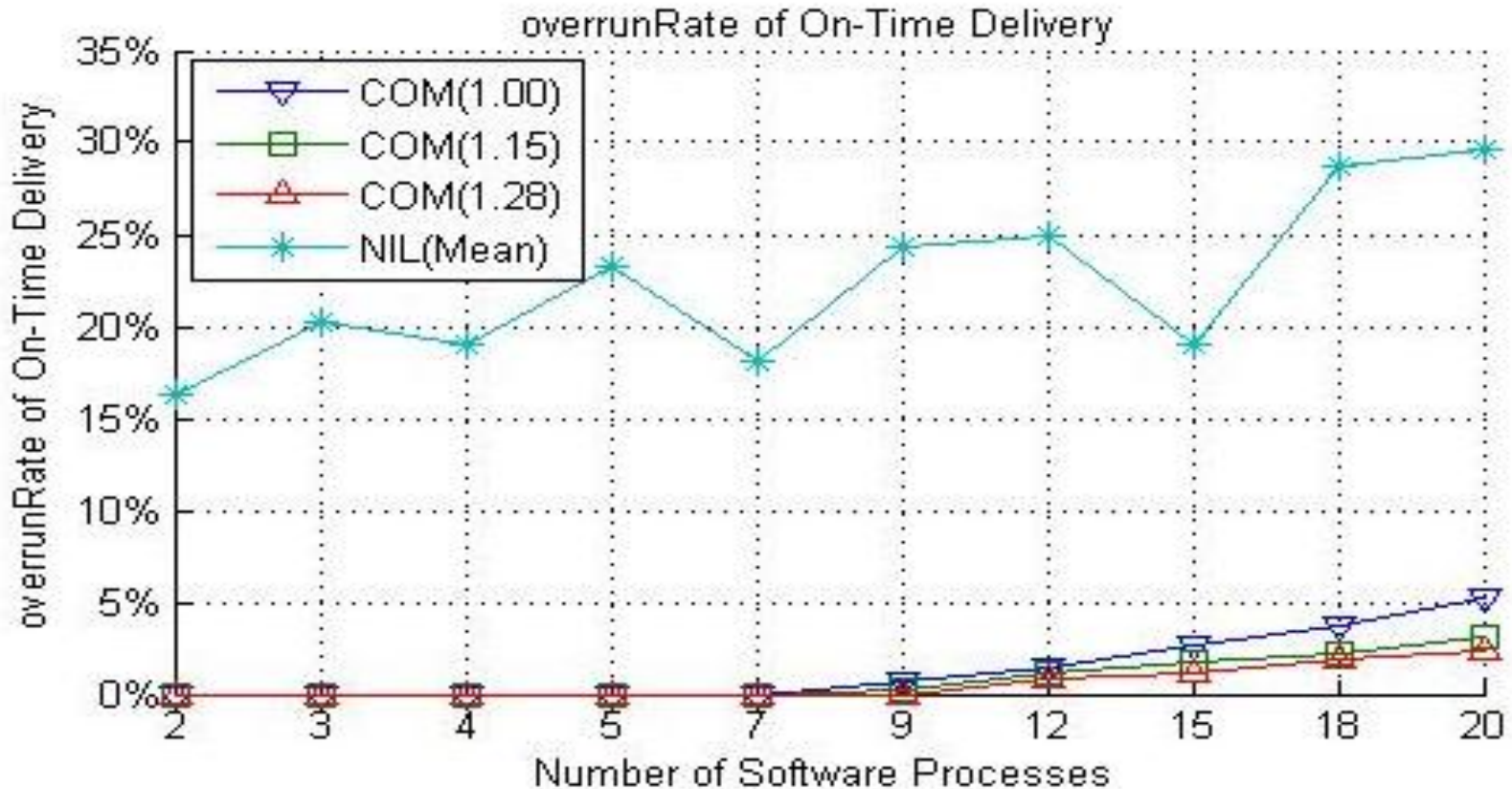
■ Two measuring attributes

- *improvedPercent* : the difference of the overall completion time before and after our GA based scheduling strategy divided by the one before
- *overrunRate*: the number of processes which fails to be completed within deadlines divided by the number of total processes

Simulation Results



Simulation Results





Where Are We

■ Introduction

- Uncertainty of Software Processes
- On-Time Delivery

■ A Two-Stage Probabilistic Scheduling Strategy

- Motivation and Strategy Overview
- Stage 1: Setting deadlines
- Stage 2: GA based project scheduling

■ Evaluation

■ Conclusion

Conclusion



- This paper proposes a two-stage probabilistic scheduling strategy which integrates statistic based schedule estimation and stochastic project scheduling in order to achieve on-time delivery of software projects.
 - Stage 1: Setting deadlines for individual software processes based on win-win negotiations
 - Stage 2: GA based project scheduling which optimises the overall completion time given real-world heuristic rules
- The simulation results demonstrate its effectiveness
- Future work
 - To tackle the case where the best scheduling plan can not be found, we will try to identify the key software processes where on-time delivery can be achieved with the minimum increase of extra cost
 - To trial in some real world software projects

The End



■ *Thanks! Any Questions?*

