[simula . research laboratory]

Experiments on Object-Oriented Analysis and Design

Erik Arisholm

[simula . research laboratory]

Series of experiments forming one quasi-experiment

- A series of five controlled experiments (can be considered as *one* quasi-experiment) where the subjects consisting of
  - 295 junior/intermediate/senior Java consultants from Norway, Sweden and the UK, and
  - 273 undergraduate/graduate students from Norway and Canada
- performed maintenance tasks on two alternative designs of the same Java system
- to assess the effects of (combinations of)
  - control style (centralized vs delegated),
  - maintenance task order (easy vs difficult task first),
  - documentation (UML versus no UML), and
  - development process (pair programming vs individual)
- on software maintainability (change effort and correctness)

[simula . research laboratory]

The individual experiments

Exp 1: The Original Experiment with students and pen-and-paper tasks (Fall 1999)

- **Effect of Centralized (bad) vs Delegated (good) Control Style**
- E. Arisholm, D. I. K. Sjøberg and M. Jørgensen. Assessing the Changeability of two Object-Oriented Design Alternatives - a Controlled Experiment, *Empirical Software Engineering* 6(3):231-277, 2001.
- 36 undergraduate students and 12 graduate students

Exp 2: The Control-style experiment with professional Java developers and Java tools (Fall 2001- Spring 2002)

- **Effect of Centralized vs Delegated Control Style for Categories of Developer**
- E. Arisholm and D. I. K. Sjøberg. Evaluating the Effect of a Delegated versus Centralized Control Style on the Maintainability of Object-Oriented Software, *IEEE Transactions on Software Engineering* 30(8):521-534, 2004
- 99 professionals, 59 students

Exp 3: UML experiments (Spring 2003 - Fall 2004)

- **Effect of UML (vs No UML) for the Delegated Control Style**
- E. Arisholm, L. C. L. Briand, S. E. Hove and Y. Labiche. The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation, *IEEE Transactions on Software Engineering* 32(6):365-381, 2006.
- 20 students from UiO (Spring 2003) + 78 students from Carleton Univ., Canada (Fall 2004)

Exp 4: Task Order Experiment (Fall 2001-Spring 2005)

- **Effect of Task Order and Centralized vs Delegated Control Style**
- A. I. Wang and E. Arisholm. The Effect of Task Order on the Maintainability of Object-Oriented Software, *To appear in Information and Software Technology*, 2008.
- **Easy task first**: 59 students from Exp 2 (2001-2002)
- **Difficult task first**: 66 students from NTNU (Spring 2005)

Exp 5: Pair programming experiment (Fall 2003-Spring 2005):

- **Effect of Pair Programming (vs individual programming) and Delegated vs Centralized Control Style**
- E. Arisholm, H. E. Gallis, T. Dybå and D. I. K. Sjøberg. Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise, *IEEE Transactions on Software Engineering* 33(2):65-86, 2007.
- 196 professional programmers in Norway, Sweden, and UK (98 pairs)
- 99 individuals (from Exp 2)

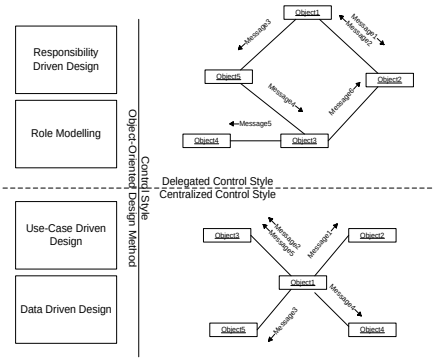
[simula . research laboratory]

A quasi-experiment of increasing scope

		Exp1	Exp2	Exp3	Exp4	Exp5
Research Question: Effect of …	Control Style	x	x	x	x	x
	UML			x		
	Task Order				x	
	Pair Programming					x
System	Coffee-machine	x	x	x	x	x
Dependent Variables	Duration (minutes)	x	x	x	x	x
	Correctness (%)	x	x	x	x	x
Control Style	Centralized (CC)	x	x		x	x
	Delegated (DC)	x	x	x	x	x
Documentation	No UML			x		
	Some UML	x	x		x	x
	Complete UML			x		
Task Order	Easy First	x	x	x		x
	Difficult first				x	
Development tools	Java Pen & Paper	x				
	Java IDE		x	x	x	x
	UML Tool			x		
Development process	Individual	x	x	x	x	
	Pair Programming					x
Subjects	BSc-students	36	27	98		
	MSc-students	12	32		66	
	Juniors		31			50
	Intermediates		32			70
	Seniors		36			76

Centralized vs Delegated Control Style

- The Delegated Control Style:
  - Rebecca Wirfs-Brock: *A delegated control style ideally has clusters of well defined responsibilities distributed among a number of objects. To me, a delegated control architecture feels like object design at its best…*
  - Alistair Cockburn: *[The delegated coffee-machine design] is, I am happy to see, robust with respect to change, and it is a much more reasonable "model of the world."*
- The Centralized Control Style:
  - Rebecca Wirfs-Brock: *A centralized control style is characterized by single points of control interacting with many simple objects. To me, centralized control feels like a "procedural solution" cloaked in objects…*
  - Alistair Cockburn: *Any oversight in the "mainframe" object (even a typo!) [in the centralized coffee-machine design] means potential damage to many modules, with endless testing and unpredictable bugs.*



5

The treatments

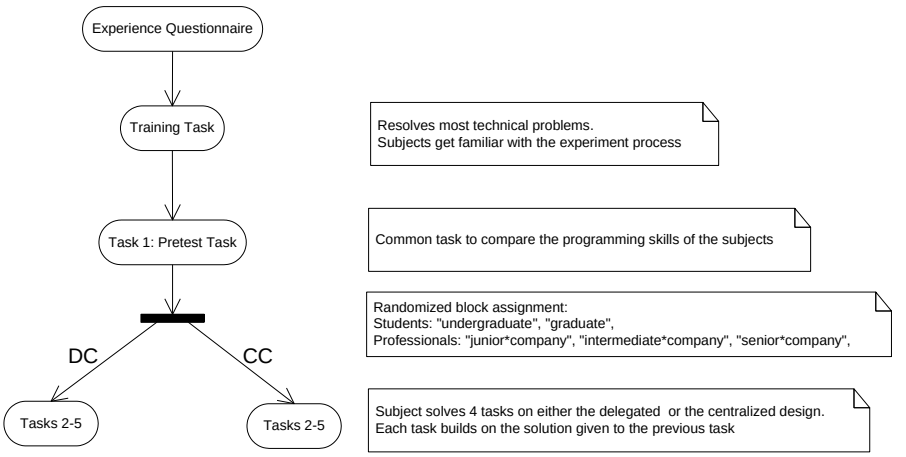
	CC	DC
CoffeeMachine	Initiates the machine; knows how the machine is put together; handles input	Initiates the machine; knows how the machine is put together; handles input
CashBox	Knows amount of money put in; gives change; answers whether a given amount of credit is available.	Knows amount of money put in; gives change; answers whether a given amount of credit is available.
FrontPanel	Knows selection; knows price of selections, and materials needed for each; coordinates payment; knows what products are available; knows how each product is made; knows how to talk to the dispensers.	Knows selection; coordinates payment; delegates drink making to the Product.
Product		Knows its recipe and price.
ProductRegister		Knows what products are available.
Recipe		Knows the ingredients of a given product; tells dispensers to dispense ingredients in sequence.
Dispensers	Controls dispensing; tracks amount it has left.	Knows which ingredient it contains; controls dispensing; tracks amount it has left.
DispenserRegister		Knows what dispensers are available
Ingredient.		Knows its name only.
Output	Knows how to display text to the user.	Knows how to display text to the user.
Input	Knows how to receive command-line input from the user	Knows how to receive command-line input from the user
Main	Initializes the program	Initializes the program

6

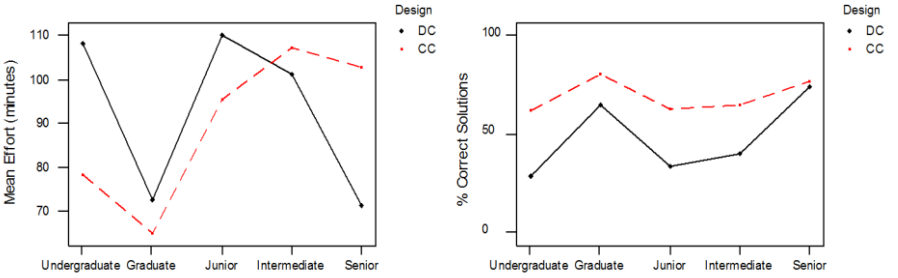
Assignment of subjects (Exp 2): Randomized Block Design

	CC	DC	Total
Undergraduate	13	14	27
Graduate	15	17	32
Junior	16	15	31
Intermediate	17	15	32
Senior	17	19	36
Total	78	80	158

Experiment design (Exp 2)



Results



DC = Delegated Control Style
CC = Centralized Control Style

The effect of control style depends mainly on the experience of the developers!

Discussion point (Exp 2)

- What are the main threats to validity of this study?
- Is it possible to extend the design to address some of the threats?

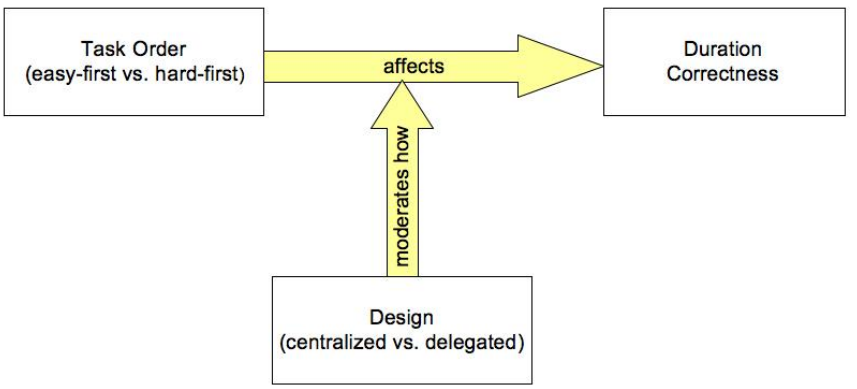
Experiment 4: The Effect of Task Order on the Maintainability of Object-Oriented Software (*)

- Research questions
  - RQ1: Does the order in which you perform maintenance tasks affect maintainability?
  - RQ2: Does the effect of task order depend on how the system is structured (control style)?

* Wang & Arisholm, The Effect of Task Order on the Maintainability of Object-Oriented Software, To appear in *Information and Software Technology (IST),* 2008.

11

Another way to look at RQ1 and RQ2

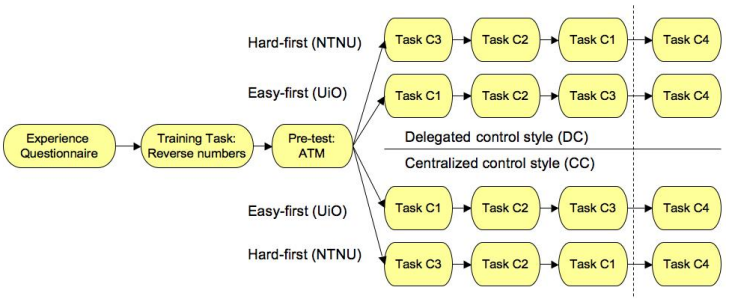


12

Hypotheses

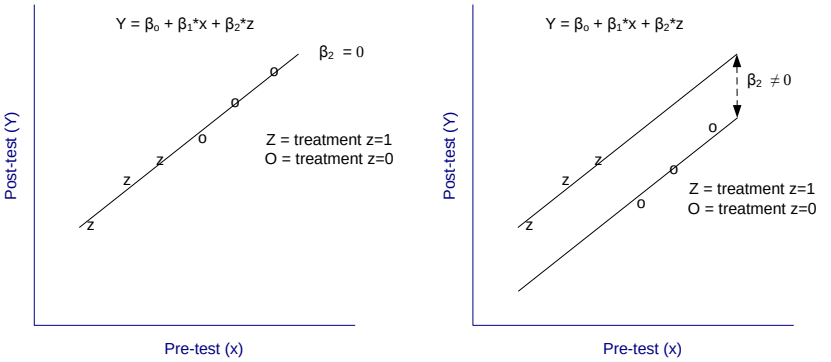
- $H0_1$ – **The Effect of Task Order on Duration.** The time taken to perform change tasks is equal for the easy-first and hard-first task order.
- $H0_2$ – **The Moderating Effect of Design on Duration.** The difference in the time taken to perform change tasks for easy-first and hard-first task order does not depend on design.
- $H0_3$ – **The Effect of Task Order on Correctness.** The correctness of the maintained programs is equal for easy-first and hard-first task order.
- $H0_4$ – **The Moderating Effect of Design on Correctness.** The difference in the correctness of the maintained programs for easy-first and hard-first task order does not depend on design.

Experiment design



Dealing with non-equivalent groups (to compare results across experiments)

Necessary to have a common pre-test to adjust for skill differences between groups (using ANCOVA)*

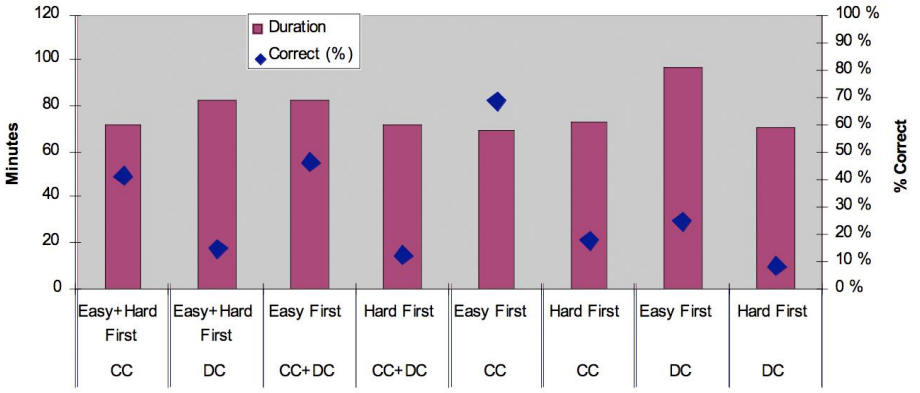


•*T.D. Cook, and D.T. Campbell (1979), *Quasi-Experimentation: Design & Analysis Issues for Field Settings*, Houghton Mifflin Company.

ANCOVA model specifications

Model	Response	Distrib.	Link	Model Term	Primary use of model term
(1)	Duration	Gamma	Log	Log(pre_dur)	Covariate to adjust for individual skill differences
				TaskOrder	**Test $H0_1$ (Duration Main Effect)**
				Design	Models the effect of design on duration
				TaskOrder x Design	**Test $H0_2$**
(2)	Correctness	Binomial	Logit	Log(pre_dur)	Covariate to adjust for individual skill differences
				TaskOrder	**Test $H0_3$ (Correctness Main Effect)**
				Design	Models the effect of design on correctness
				TaskOrder x Design	**Test $H0_4$**

Effects of Task Order and Control Style



Discussion points (Exp 4)

- What are the main threats to validity of this experiment?
- What are the practical implications of the results?

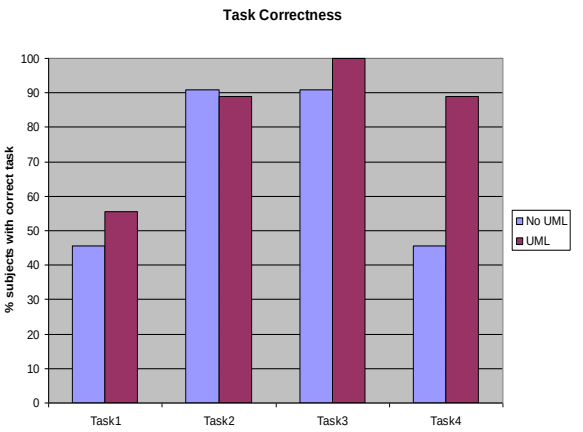
Exp. 3: Does complete UML documentation with use-case, sequence and class-diagrams help novice developers (BSc-students) to understand the delegated control style?

See also:
E. Soloway, R. Lampert, S. Letowski, D. Littman, and J. Pinto, "Designing documentation to compensate for delocalized plans," *Communications of the ACM, vol. 31, pp. 1259 - 1267, November 1988.*

19

[simula . research laboratory]

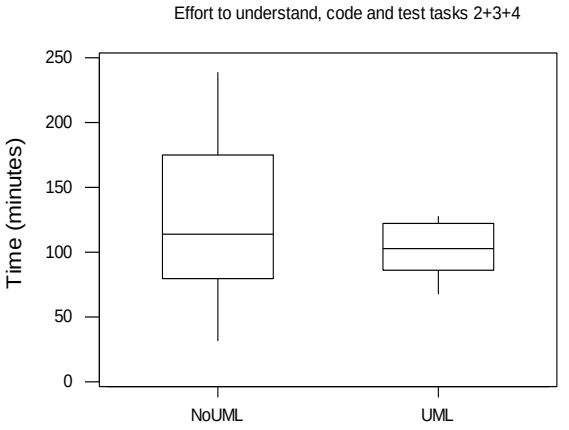
Task correctness with or without UML



•* E. Arisholm, L. C. Briand, S. E. Hove and Y. Labiche. The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation, *IEEE Transactions on Software Engineering* 32(6):365-381, 2006.

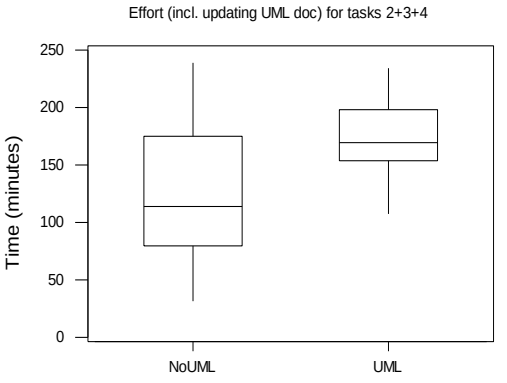
[simula . research laboratory]

Effort spent on coding and testing



[simula . research laboratory]

Effort spent on coding, testing and updating the diagrams*



Discussion points (Exp 3)

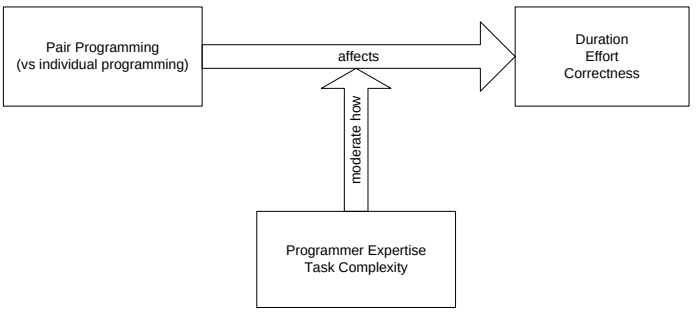
- What are the main threats to validity of this experiment?
- How would you extend the experiment to address some of these threats?

The Pair Programming Experiment (Exp 5)

- Independent variable:
  - Pair programming vs. individual programming
- Moderator variables:
  - Developer category (junior, intermediate, senior)
  - Control-style (delegated and centralized)
- Dependent variables:
  - duration, effort and correctness
- Treatments:
  - Junior, intermediate and senior pairs and individuals were randomly assigned to one of two alternative (delegated or centralized) designs of a coffee-machine, and performed four (incremental) maintenance tasks on the system
- Subjects:
  - 295 junior, intermediate and senior professional Java consultants:
    - 99 individuals (conducted in 2001/2002)
    - 98 pairs (conducted in 2004/2005)

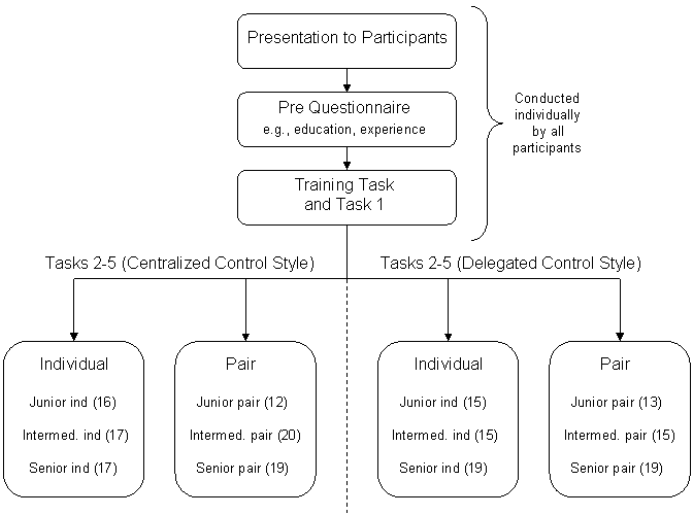
Research Question

What is the effect (regarding duration, effort and correctness) of pair programming (vs. individual programming) for various levels of programmer expertise and task complexity when performing change tasks?



25

Quasi-Experimental Design (combines subjects from Exp 2 and Exp 5)



26

Some of the preparations…

- Initial pilot experiment (presented at ISERN'02 in Japan)
- Design of the main experiment (presented at ISERN'03 in Italy)
- Quality assurance of experimental materials (three pilot experiments)
- Defining and evaluating questionnaires (pre-, post-, and personal characteristics)
  - More than 200 questions
  - Tested the questionnaires on 28 professional developers
- Translating all experimental materials to Swedish and English (from Norwegian)
- Development of automatic test-script and web-based review system
  - Two independent reviewers analyzed all task solutions manually (more than 1000 tasks)
- Collaboration with a psychologist, who was responsible for the administration and analysis of a "big five" personality test
- Recruiting 295 Java consultants from 27 consultancy companies in Norway, Sweden and UK!

Formal Hypotheses

$H0_1$	**Effect of Pair Programming on Duration:** The duration to perform change tasks is equal for individuals and pairs.
$H0_2$	**Effect of Pair Programming on Change Effort:** The effort spent to perform change tasks is equal for individuals and pairs.
$H0_3$	**Effect of Pair Programming on Quality:** The quality of the maintained programs is equal for individuals and pairs.
$H0_4$	**Moderating Effect of Programmer Expertise on Duration:** The difference in the duration to perform change tasks for pairs versus individuals does not depend on programmer expertise.
$H0_5$	**Moderating Effect of Programmer Expertise on Change Effort:** The difference in the effort spent to perform change tasks for pairs versus individuals does not depend on programmer expertise.
$H0_6$	**Moderating Effect of Programmer Expertise on Quality:** The difference in the quality of the maintained programs for pairs versus individuals does not depend on programmer expertise.
$H0_7$	**Moderating Effect of Task Complexity on Duration:** The difference in the duration to perform change tasks for pairs versus individuals does not depend on task complexity.
$H0_8$	**Moderating Effect of Task Complexity on Change Effort:** The difference in the effort spent to perform change tasks for individuals and pairs does not depend on task complexity.
$H0_9$	**Moderating Effect of Task Complexity on Quality:** The difference in the quality of the maintained programs for pairs versus individuals does not depend on task complexity.

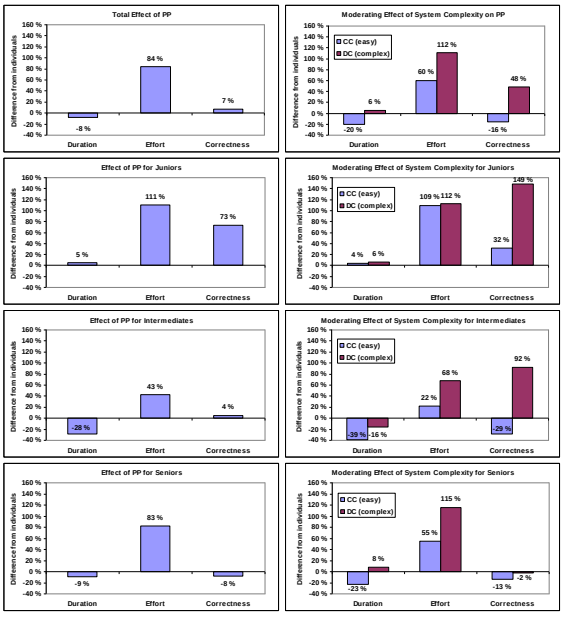
ANCOVA model specifications

Model	Response	Model Term	Primary use of model term
(1)	Log(Duration)	Log(pre_dur)	Covariate to adjust for individual differences in programming skill
		PP	**Test H01 (Duration Main Effect)**
		CS	Assess the effect of control style on duration
		Log(pre_dur)*PP	**Test H04**
		Log(pre_dur)*CS	Test for ACOVA assumption of homogeneity of slopes
		PP*CS	**Test H07**
		Log(pre_dur)*PP*CS	Test for ACOVA assumption of homogeneity of slopes
(2)	Log(Effort)	Log(pre_dur)	Covariate to adjust for individual differences in programming skill
		PP	**Test H02 (Effort Main Effect)**
		CS	Assess the effect of control style on duration
		Log(pre_dur)*PP	**Test H05**
		Log(pre_dur)*CS	Test for ACOVA assumption of homogeneity of slopes
		PP*CS	**Test H08**
		Log(pre_dur)*PP*CS	Test for ACOVA assumption of homogeneity of slopes
(3)	Logit(Correctness)	Log(pre_dur)	Covariate to adjust for individual differences in programming skill
		PP	**Test H03 (Correctness Main Effect)**
		CS	Assess the effect of control style on duration
		Log(pre_dur)*PP	**Test H06**
		Log(pre_dur)*CS	Test for ACOVA assumption of homogeneity of slopes
		PP*CS	**Test H09**
		Log(pre_dur)*PP*CS	Test for ACOVA assumption of homogeneity of slopes

29

[simula . research laboratory]

Results



14/11/2005 30

Summary of results

- Performing change tasks on a delegated control style requires, on average, more time and results in more defects than on a centralized control style, in particular for novices (undergraduate students and junior consultants)
  - Only seniors seem to have the necessary skills to benefit from the more ”elegant” delegated control style.
  - Explanation: Unlike experts, novices perform a mental trace the code in order to understand it. This tracing effort is more difficult in a delegated control style
  - Results are consistent when performing the most difficult task first (though the disadvantage of a delegated control style is smaller than for easy first)
- Two ways to *decrease* the cognitive complexity of the delegated control style (for novices in particular)
  - Extensive UML documentation
  - Pair programming