



THE UNIVERSITY
OF QUEENSLAND

CSSE2003

Software Engineering Studio

Semester 2, 2009

1: Introduction and Time Monitoring

CSSE2003 People

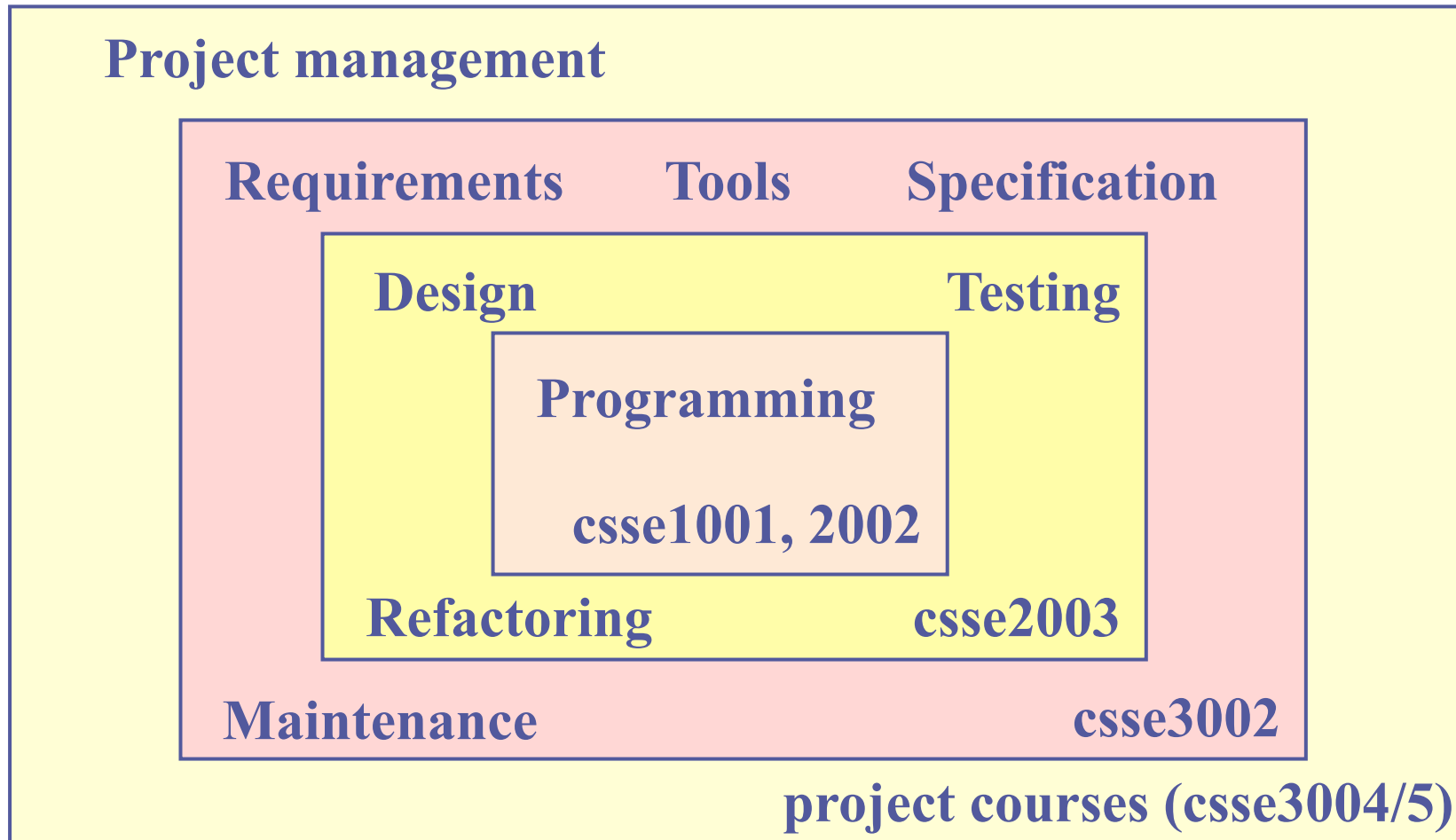
- ◆ Coordinator: David Carrington
- ◆ Lecturers: Jorn Guy Suess

- ◆ Tutors: Joshua Bartlett
Xin Wang (John)

Lecture Summary

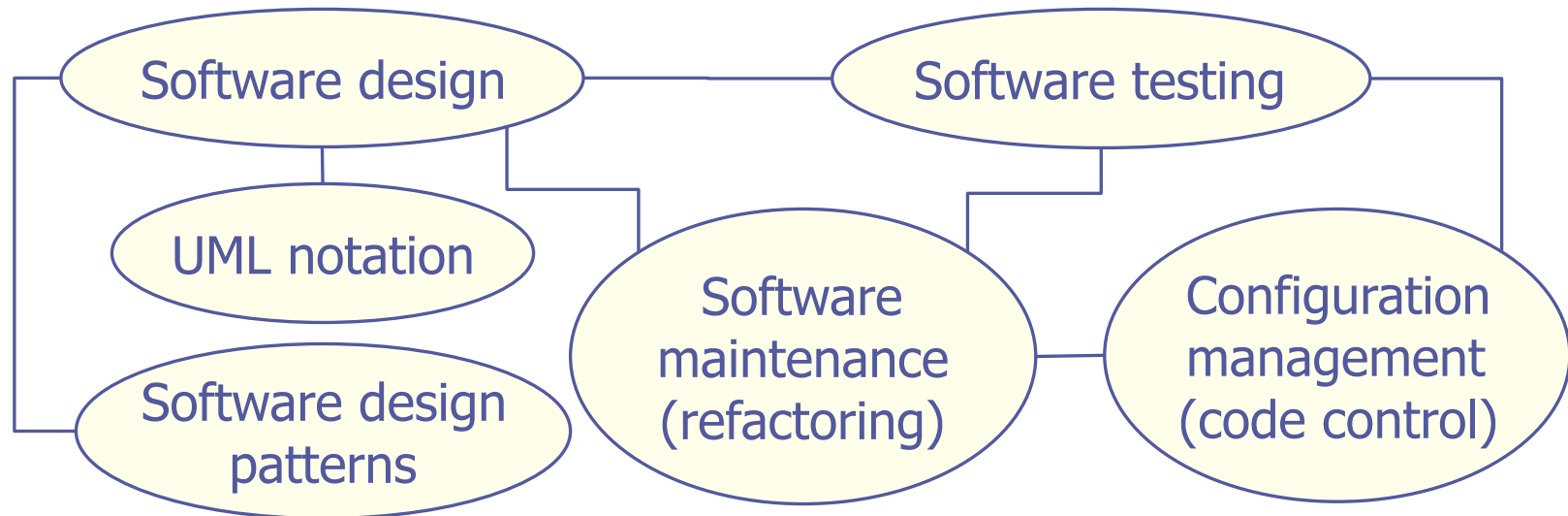
- ◆ Course introduction
 - goals
 - assumed background
 - motivation, teaching & learning philosophy
 - expected outcomes
 - teaching activities & resources
 - assessment & grading
- ◆ Time Monitoring
- ◆ Survey

CSSE2003 in context



Course Goals

- ◆ Demonstrate concepts and practice of:



- ◆ Extend students' experience in these areas
- ◆ Provide students with positive experiences of collaborative learning and some appreciation of the need for life-long learning skills.

Assumed Background

- ◆ Prerequisite: CSSE2002, COMP2500
- ◆ Incompatible: COMP2801, COMP2501
- ◆ Students are expected to have:
 - some experience programming in Java
 - an interest in extending their software development experience

Expected Outcomes - 1

- ◆ Software design:
 - understand and apply design patterns
 - Represent the design of a software product using the UML notation
 - document the design decisions underlying a software design.
- ◆ Software maintenance:
 - apply refactoring strategies to improve the quality of existing designs and code.
 - manage an evolving software product using configuration management and software build tools.

Expected Outcomes - 2

- ◆ Testing:
 - develop sets of test cases and choose appropriate test strategies.
- ◆ Collaborative learning:
 - Participate effectively as a member of a software development team.

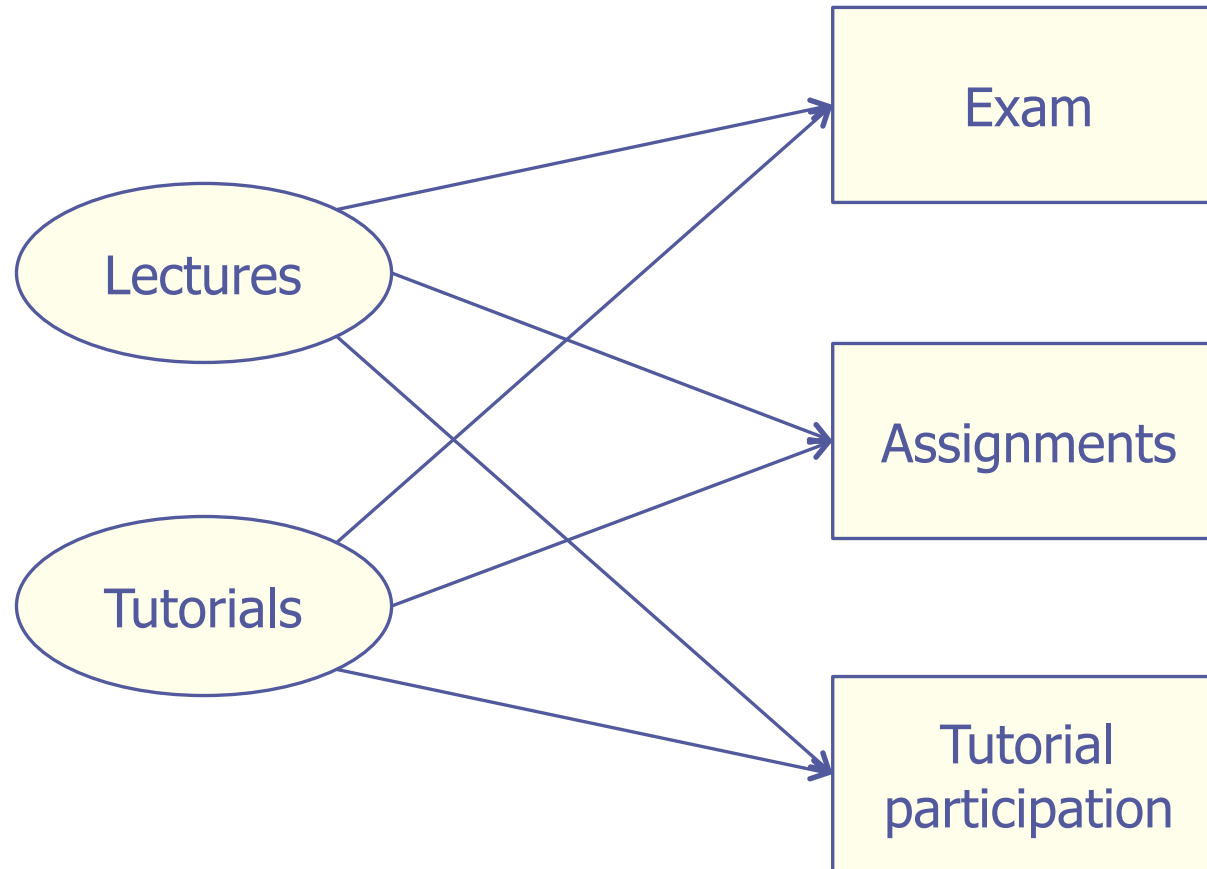
Motivation

- ◆ Software design transforms a software specification into a structure suitable for implementation.
- ◆ Design increases in importance with the size and structure of problems we tackle.
- ◆ Testing is a necessary development activity and needs to be applied rigorously and systematically.

Teaching and Learning Philosophy

- ◆ Rather than education as transmission of information from teacher to learner, we believe in active learning.
- ◆ Our task is to create a pathway and guide students through a series of learning activities that promote new knowledge and skills.
- ◆ This course requires significant team work, which may initially be difficult for some students.
- ◆ Ability to work effectively in teams is consistently rated as one of the most important skills for the workplace.

Teaching activities & Assessment



Teaching Activities - Lectures

- ◆ Wednesday 12 – 2 pm room 50-S201

- ◆ Lectures
 - explain the principles underlying the course content,
 - introduce software engineering techniques, and
 - present examples to help students understand the principles and techniques.

Teaching Activities - Tutorials

- ◆ Each student needs to register for one tutorial using **Signon** (on SI-net) – open until 5pm Friday.
 - After that, see me!
- ◆ All tutorials are held in 78-208.
- ◆ Teams for all three group assignments will be formed in the tutorial classes in week 2.

Please arrive on-time!

Teams and Assignments

- ◆ Team assignments are done in teams of 3 or 4 (preferred).
- ◆ Teams must attend the **same** tutorial class.
- ◆ Student may form their own teams, but we may have to re-arrange these to satisfy the needs of the whole class, e.g. adding an extra member to a team of 3.
- ◆ Tutors will organise team membership in the week 2 tutorial.
- ◆ The mark for each assignment will be distributed (with some constraints) to members by nominating the percentage contribution for each member.

Team Assignments

- ◆ The team assignments are coordinated stages in an investigation of an open source software product:
 - A1: selection, use, version control of code
=> *tutorial presentation*
 - A2: investigation of internal design structures
=> *report*
 - A3: enhancement and/or extension
=> *report and tutorial presentation (including enhancement demonstration)*

Why Open Source Software?

- ◆ Challenges:
 - “helping students understand the differences between the small programs that they write as educational exercises and the large scale (real) software products that they will deal with when they are working...
 - Software engineering education need techniques to help their students cross this gap as they transition from novices to practitioners...”
- ◆ Studying open source software provides SE students with realistic and challenging examples

from David Carrington, What can Software Engineering Students Learn from Studying Open Source Programs?, Int. J. Engineering Education., 2008

Individual Assignments

- ◆ Time monitoring
=> *submission on the web*
- ◆ Time monitoring reflection
=> *report*
- ◆ Active tutorial participation
=> *assessed by tutors in the lab*
- ◆ Programming and testing
(code review, test case design, fixing errors in code)
=> *code and test cases*

Teaching Plan

- ◆ Refer to the course profile - it will change, but just how is not yet determined!
- ◆ List of topics:
 - Version Control of Code
 - Unified Modeling Language (UML)
 - Software design patterns
 - Refactoring
 - Object-oriented testing
 - Software architecture

Resources

- ◆ Web: www.itee.uq.edu.au/~csse2003
- ◆ Newsgroup: uq.itee.csse2003
- ◆ Notes: on web (in PDF)

Required Text Book

- ◆ Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd Edition, Prentice Hall, 2004.
 - Earlier editions of this text are acceptable although students should realise that references will be to the latest edition, which uses UML2.
 - This text is also a text for CSSE3002.
- ◆ Many other reference books are listed in the course profile, and are available in the PSE library.

Assessment

	<u>Weight</u>
◆ 3 team assignments:	
• A1: tutorial presentation	5%
• A2: initial written report	15%
• A3: final report/presentation	20%
◆ Tutorial participation	8%
◆ Time monitoring	7%
◆ Programming and testing	10%
◆ Final exam	35%
• 2 hours plus 10 mins perusal	
• open book	

Grading

Grade	Total	Constraint
1	0 - 19 %	
2	20 - 44 %	
3	45 - 49 %	Exam \geq 40 %
4	50 - 64 %	Exam \geq 45 %
5	65 - 74 %	Exam \geq 60 %
6	75 - 84 %	Exam \geq 70 %
7	85 - 100 %	Exam \geq 80 %

Time Monitoring

- ◆ You are required to monitor the amount of time you allocate to **all your courses**. Each week, you are to report three values associated with **just** CSSE2003 activities:
 - in-class time,
 - solo time, and
 - team time.
- ◆ After week 10, you are to complete a one to two page report on the time data collected, summarising the time that you allocated to **each course**, including a self-evaluation/reflection.

How is the data to be submitted? -1

- ◆ A web page linked to the CSSE2003 home page (www.itee.uq.edu.au/~csse2003) will be used to collect data from students.
- ◆ You are required to enter the data for each week in the following week before 10pm on Wednesday.
- ◆ The reporting week is from Monday to Sunday.
- ◆ The page requires that you login (so that you get credit for your data) and enter:
 - class hours,
 - solo study hours, and
 - team study hours.

How is the data to be submitted? -2

- ◆ Fractional time values are permitted; values to the nearest half-hour are sufficiently accurate.
- ◆ Multiple submissions are accepted – only the final one before the deadline is retained.
- ◆ The review after the mid-semester break is to be submitted on-line as a PDF document – see the course profile for submission details

How are the marks allocated?

- ◆ The 14 weekly reports contribute 3%.
- ◆ The review after the mid-semester break week contributes 4% (see the sample form on page 4 for the marking scheme).
- ◆ Late submissions of time data will receive no marks unless an acceptable explanation is provided.

Why is this required?

1. To help you develop insight into how you use your time, possibly leading to improvements. Almost all time management advice suggests that keeping records is a necessary first step to improving your use of time.
2. To give me feedback on student behaviour and to allow me to give general feedback to the class.
3. The personal software process developed by Watts Humphrey (see his books: *A Discipline for Software Engineering* or *Introduction to the Personal Software Process*) requires detailed recording and analysis of activities and software defects.

Why is it assessable?

- ◆ To give you an incentive for completing this task since most people find it requires considerable discipline to monitor their time.
- ◆ To indicate that we consider this task important and worthwhile.

How will the data be used?

- ◆ To generate class distributions on a weekly basis as feedback so that you can see how your time usage compares to your peers.

Why do students have to identify their data?

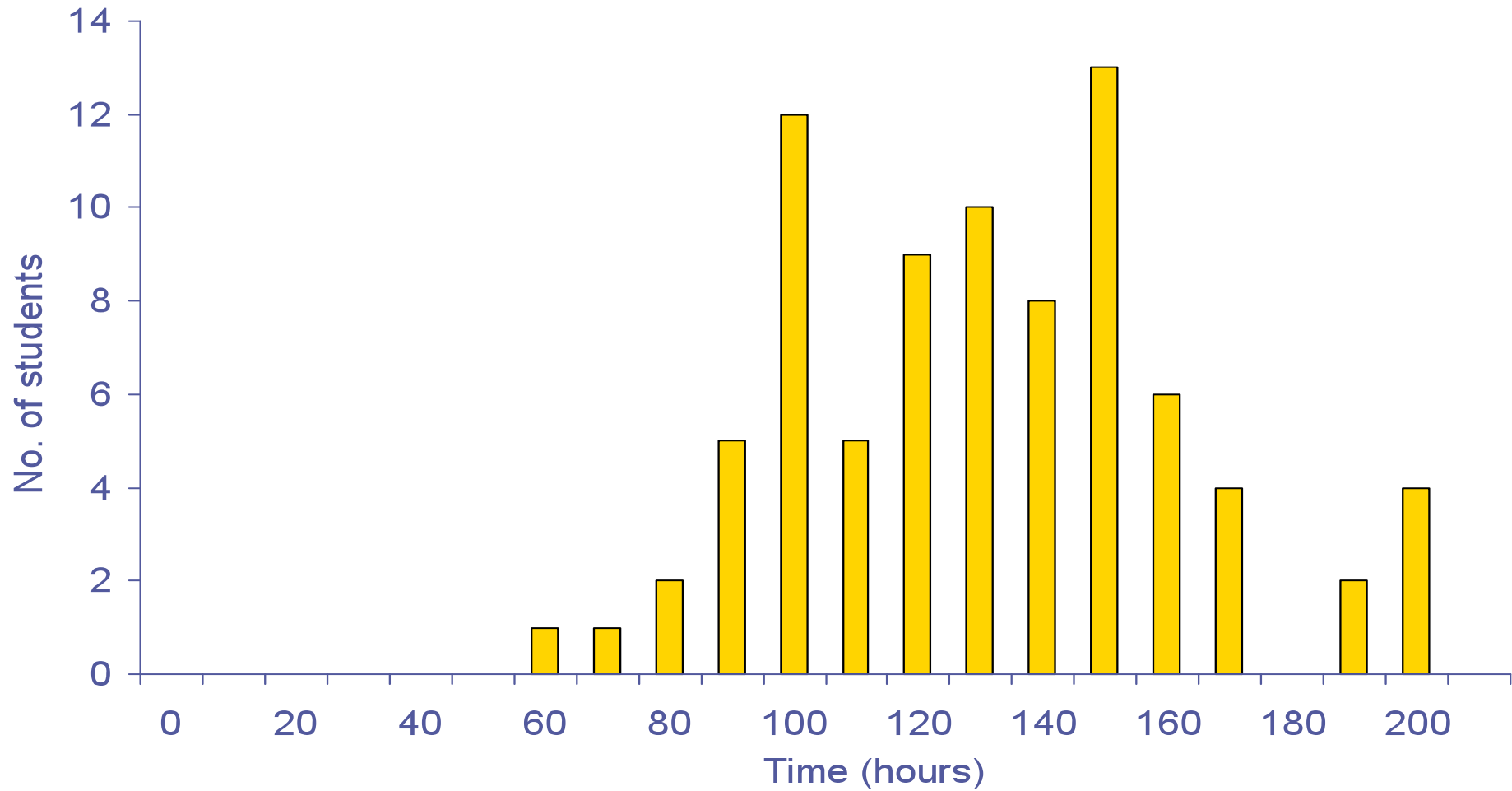
1. To allow marks to be allocated.
2. To encourage people to be responsible with the data they supply. The actual data values do not influence the mark, but some consistency checks will be applied.

Actual or made-up data?

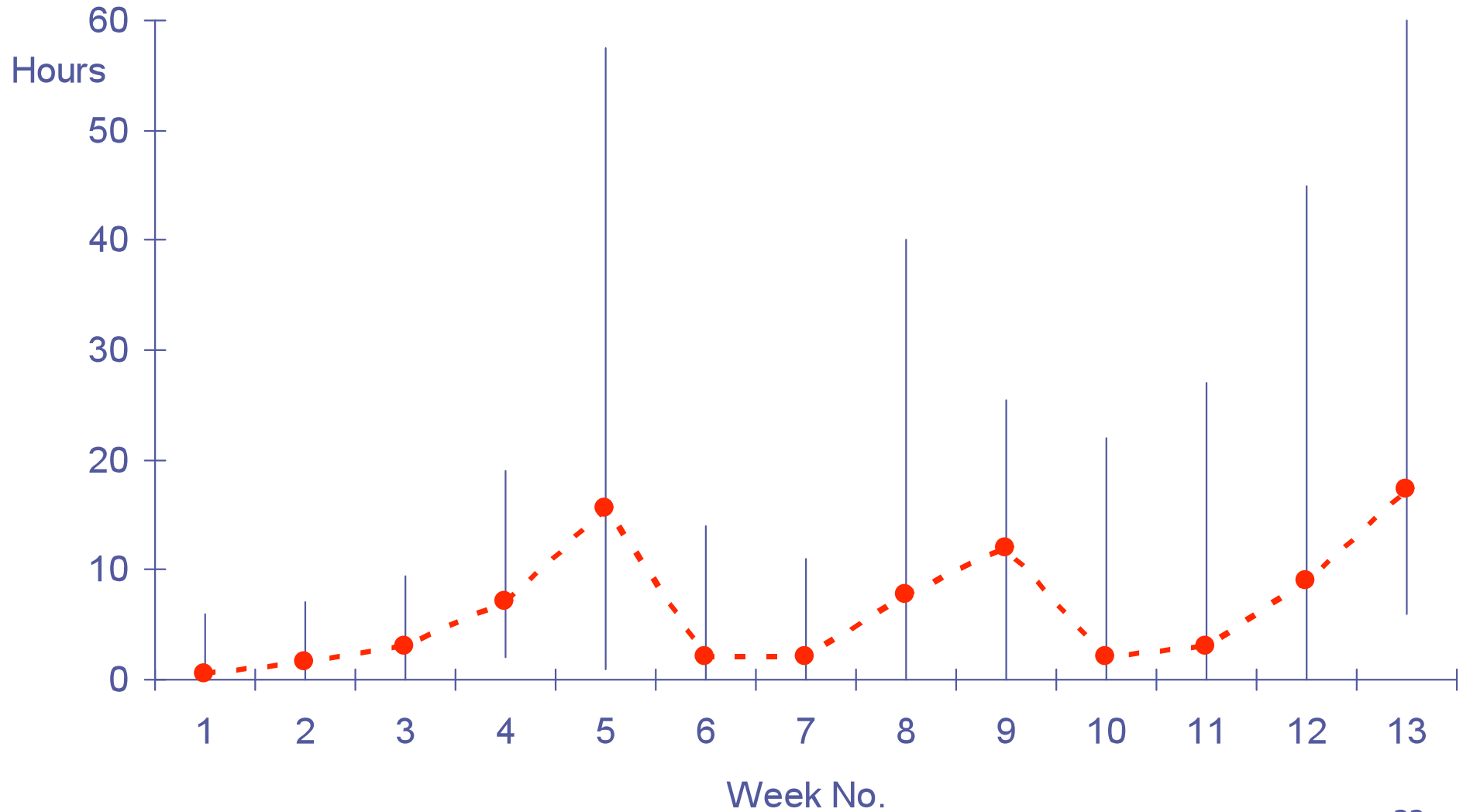
Suggestions

1. Copy the supplied template to record your activities as you go. It is much easier to remember the last few hours than a full week.
2. You need to record all your courses to get a global picture of your use of time.
3. One suggestion is to enter in each cell of the template designations like "2003C" and "2003S" where "C" is for class, "S" is for solo study and "T" is for team work.
4. At the end of each week, count the number of occurrences of each letter for CSSE2003 (dividing by 2 to get hours).

Results: Time for semester



Results: Weekly



Time Monitoring

Any Questions?

Survey

- ◆ The survey gives me some information about you.
- ◆ We try to use the results to influence how the course is run.
- ◆ The survey is anonymous so feel free to be honest in your feedback.

Action List

1. Register for your tutorial class if you have not already done so.
2. Consider forming a team in advance – must be in same tutorial class.
3. Attend your tutorial class next week (in 78-208).
4. Start time monitoring from today and report weekly.