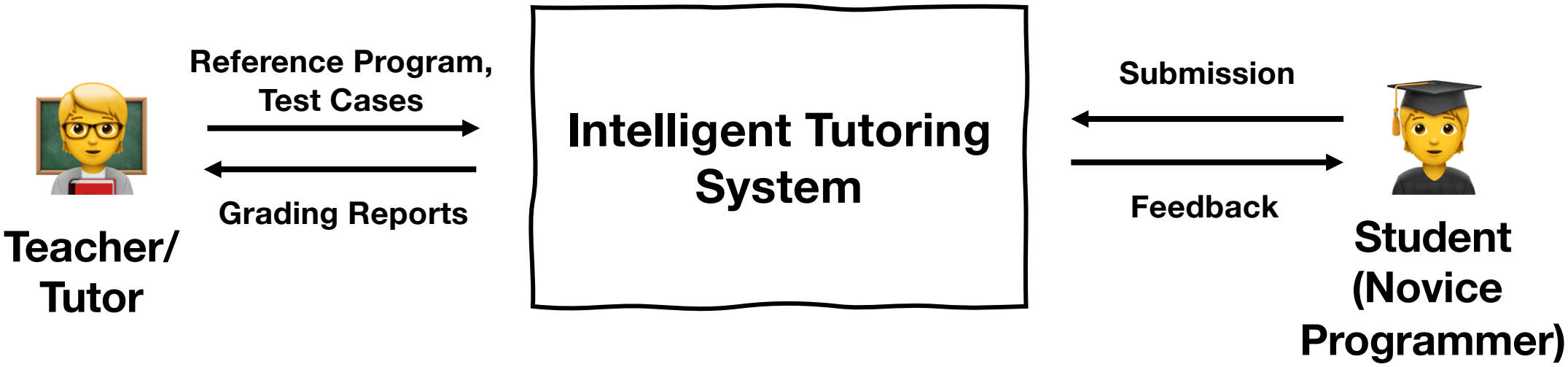
CS3213 Project – Week 1

Introduction to Course Project | 12-01-2022

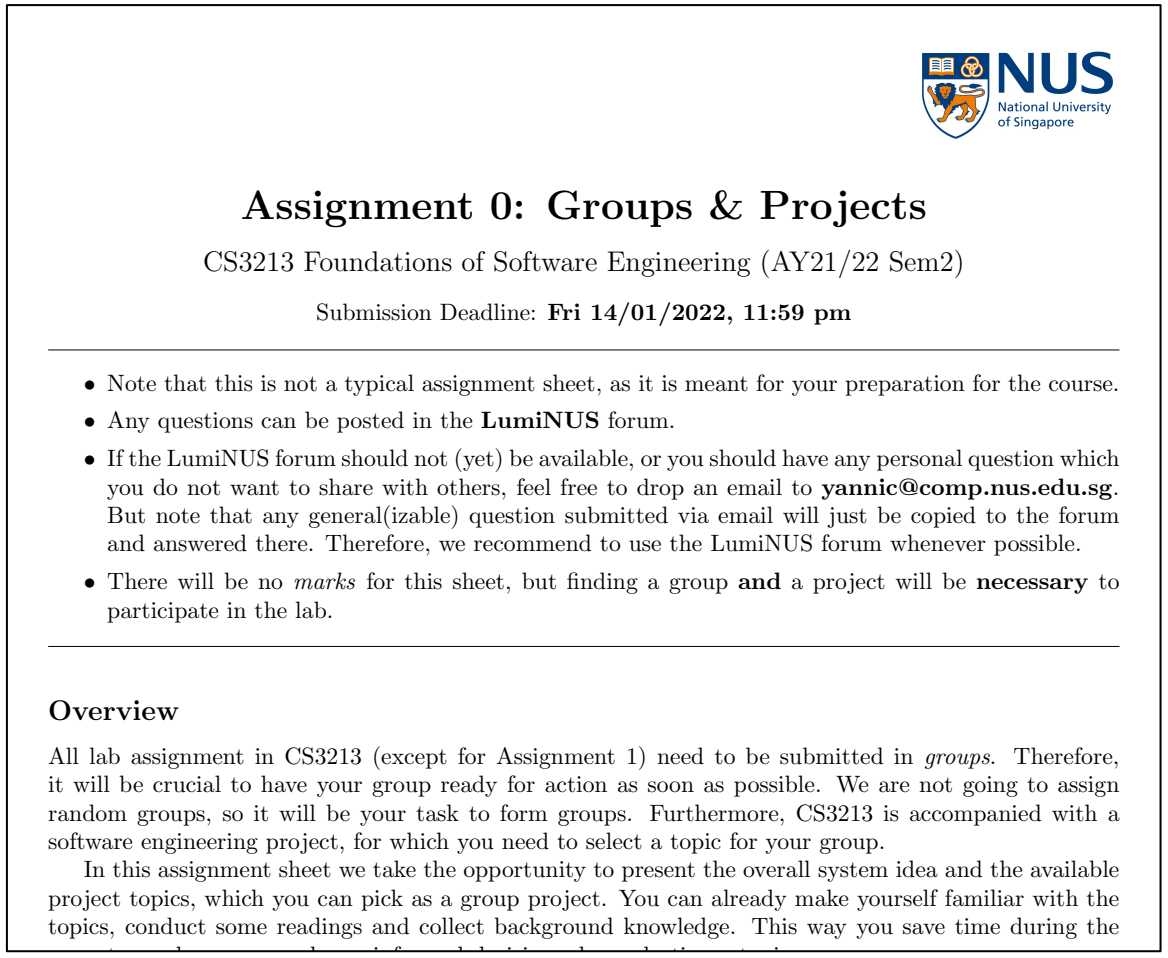
- ❑ Idea of Course Project
- ❑ Lab Logistics/Amenities
- ❑ Group Selection Process
- ❑ Assignment 1
- ❑ Requirement Elicitation in Week 2

Idea of the Course Project

- The **lecture** provides software engineering methods, theory, models, patterns, etc.
- The **lab** provides application of the software engineering principles.
- Not just any project but related to our **research** including **real customers** at NUS.



Course Project - Topics



NUS National University of Singapore

Assignment 0: Groups & Projects

CS3213 Foundations of Software Engineering (AY21/22 Sem2)

Submission Deadline: **Fri 14/01/2022, 11:59 pm**

- Note that this is not a typical assignment sheet, as it is meant for your preparation for the course.
- Any questions can be posted in the **LumiNUS** forum.
- If the LumiNUS forum should not (yet) be available, or you should have any personal question which you do not want to share with others, feel free to drop an email to **yannic@comp.nus.edu.sg**. But note that any general(izable) question submitted via email will just be copied to the forum and answered there. Therefore, we recommend to use the LumiNUS forum whenever possible.
- There will be no *marks* for this sheet, but finding a group **and** a project will be **necessary** to participate in the lab.

Overview

All lab assignment in CS3213 (except for Assignment 1) need to be submitted in *groups*. Therefore, it will be crucial to have your group ready for action as soon as possible. We are not going to assign random groups, so it will be your task to form groups. Furthermore, CS3213 is accompanied with a software engineering project, for which you need to select a topic for your group.

In this assignment sheet we take the opportunity to present the overall system idea and the available project topics, which you can pick as a group project. You can already make yourself familiar with the topics, conduct some readings and collect background knowledge. This way you save time during the

- **Assignment 0**: Overview about topics and projects

1 Parsing

1.1 C Parser: Develop a parser to transform C programs into a (provided) common data structure based on the control-flow graph (CFG). Additionally, provide a concretizer, which back-transforms the program in the internal common data structure to a C source file.

1.2 Python Parser: Same as 1.1 for Python.

[Coding: High, Theory: Low, Research: -, HCI: -]

2 Aligning / Matching of Programs

2.1 CFG-Based Alignment: Develop an automated alignment of the reference program and the submitted program based on the basic blocks of the programs' control-flow graph (CFG) representation. This also includes the development of an automated mapping for the variables between the reference program and the submitted program.

[Coding: Medium, Theory: Medium, Research: Low, HCI: -]

3 Error Localization / Program Interpretation (1/2)

3.1 C Interpreter: Develop an interpreter that allows to execute a C program with regard to the basic blocks in its CFG. Further, use the provided test cases to identify the root cause of the problem with regard to the basic blocks in the CFG. Implement an error localization that compares the execution traces of a reference program and the submitted program.

3.2 Python Interpreter: same as 3.1 for Python

[Coding: High, Theory: Medium, Research: -, HCI: -]

3 Error Localization / Program Interpretation (2/2)

3.3 Error Localizer: Conduct a literature study on error localization. Develop at least two error localization algorithms from different domains (e.g., statistical fault localization and analysis-based fault localization) for the provided framework and evaluate their efficacy.

[Coding: Medium, Theory: High, Research: Low, HCI: -]

4 Transforming / Repairing Programs (1/3)

4.1 Refactoring-based Repair: Develop a repair workflow that first generates semantic-preserving refactorings of a reference program so that it increases the chances of a structural alignment with a submitted program (see Project 2.1). Afterwards, it uses a matching refactoring to repair the submitted program by mutating program expressions. Strive for a minimal repair which satisfies the failing test case(s).

[Coding: Medium, Theory: Medium, Research: Medium, HCI: -]

4 Transforming / Repairing Programs (2/3)

4.2 Optimization-based Repair: Develop a repair algorithm that (1) generates local repairs at each basic block by matching the submission and the reference solution, and (2) determines the complete repair (i.e., a subset of local repairs) by using some optimization strategy, which minimizes the overall repair cost.

[Coding: Medium-High, Theory: High, Research: Low, HCI: -]

4 Transforming / Repairing Programs (3/3)

4.3 Synthesis-based Repair: Develop a repair algorithm that searches for a repair by synthesizing program expressions. The synthesis will be driven by the available components at the specific source location. It requires a specification inference, which results in a repair constraint.

[Coding: Medium, Theory: High, Research: Medium, HCI: -]

5 Feedback Generation

5.1 Automated Feedback: Develop a feedback mechanism to summarize all obtained results in an appropriate and comprehensible manner for the user. For example, show root causes of the problems and provide explanation by annotating the code.

[Coding: Low, Theory: Medium, Research: Medium, HCI: High]

5.2 Automated Grading: Develop a automated grading mechanism, which is beyond simple output of passing and failing test cases, e.g., it should take into account the necessary effort for fixing the submitted program.

[Coding: Low, Theory: High, Research: High, HCI: Low]

Team

Lecture

Abhik Roychoudhury
https://www.comp.nus.edu.sg/~abhik/

Lab and project coordination

Yannic Noller
https://www.comp.nus.edu.sg/~yannic/

Tutors

Zhiyu Fan

Jon Chua

Guo Ai

Kishore R

Liu Yu

General Setup

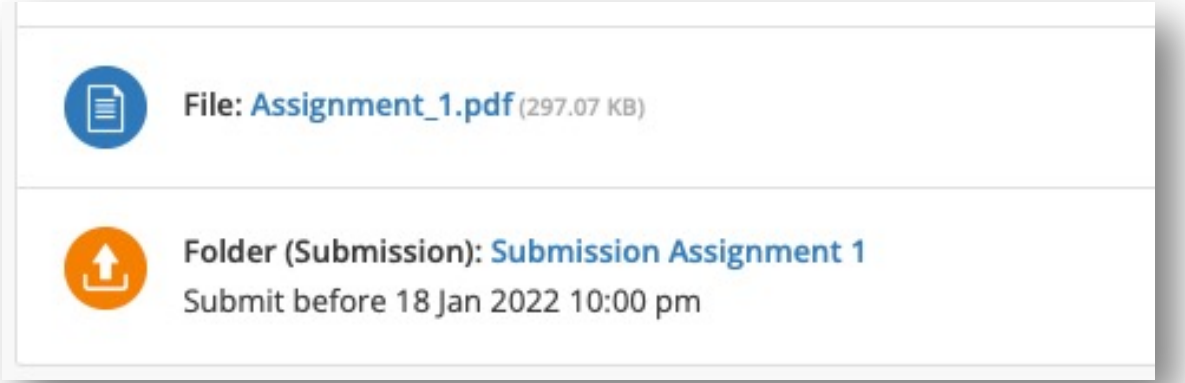
- After the 2-hour lecture slot, there is a **1-hour slot** with a **general lab session** for all students.
- Additionally, there are dedicated **tutorial sessions (aka *labs*)** with tutors with a smaller set of students.
- Until week 5, there is a **weekly assignment** sheet, which needs to be submitted by each group.
- After week 6, the focus is on project implementation.

Lab Assignments and Timeline

Week	Lab Session	Assignment	Assignment Due
1	Introduction	A1 – Requirements Analysis & Elicitation	Week 2
2	Requirements Elicitation with Customer	A2 – Requirements Modeling	Week 3
3	Requirements Modeling	A3 – Behavioral Modeling & Architectural Drivers	Week 4
4	*ITS Architecture & Projects (CNY)*	A4 – Module Design / Strategy Plan	Week 5
5	Module Design and Planning	A5 – Project Planning	Week 6
6	Unit Testing	A6 – Intermediate Deliverable	**Week 7**
R	*Recess Week*	-	-
7	Advanced Testing	-	-
8	Fault Localization and Debugging	-	-
9	Static Analysis	A7 – Test Case Design	Week 10
10	Implementation	A8 – Presentation + Final Code	**Week 12**
11	Integration Testing	-	-
12	Lab Closing Session	A9 – Final Report	**Week RD**
13	Student Presentations	-	-
RD	*Reading Week*	-	-
E	*Examination*	-	-

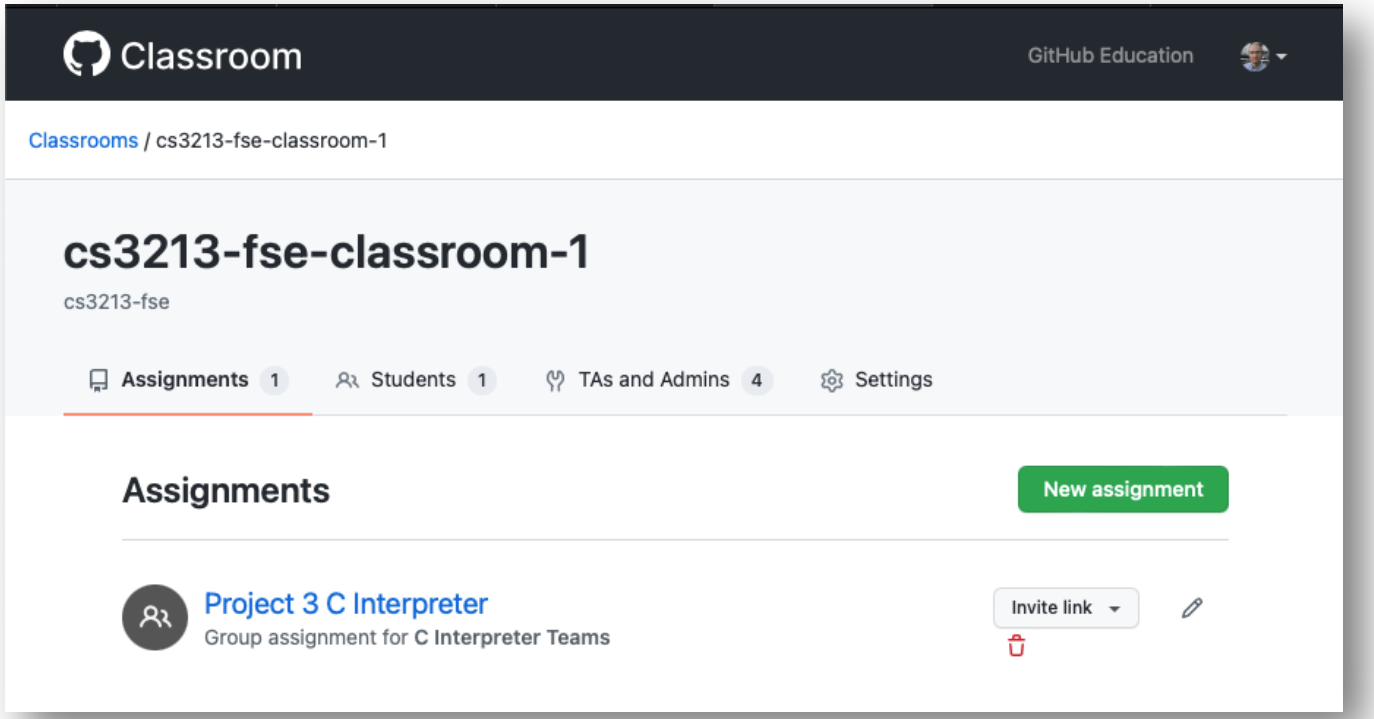
LumiNUS Submissions

- The assignment sheets and lab slides are available in LumiNUS.
- Each submission will be made by LumiNUS, except for the project code (see next slide).



Project Code Submissions

- GitHub Classroom with project templates/skeletons and integration and test pipelines.



⇨ *More details will follow in the upcoming weeks...*

Group Selection

- Each assignment (except for the first one) and the project work need to be done in submission **groups**.
- Each group will have 3-4 students.
- Coordinate with your fellow students and register in LumiNUS. We provided a Google Sheet[1] for a preliminary selection of groups and projects.

Deadline for group registration in Google Sheet:
Friday, Jan 21, 10 am

[1] https://docs.google.com/spreadsheets/d/15sk6WnvQHTjClhMi_TUuyDkylow6lOmMHVhD5n3Qu-I/edit?usp=sharing

Lab Slots Allocation

- ❏ Make sure that you and your teammates are in the same lab session!
- ❏ Labs happen Thursdays, we have six slots:

Slot Id	Time	Name
T1	10:00 – 11:00	Yannic
T2	11:00 – 12:00	Guo Ai
T3	12:00 – 13:00	Kishore
T4	13:00 – 14:00	Zhiyu
T5	14:00 – 15:00	Jon
T6	15:00 – 16:00	Jon

⇨ *The first tutorial starts in week 2, so no tutorials this week!*

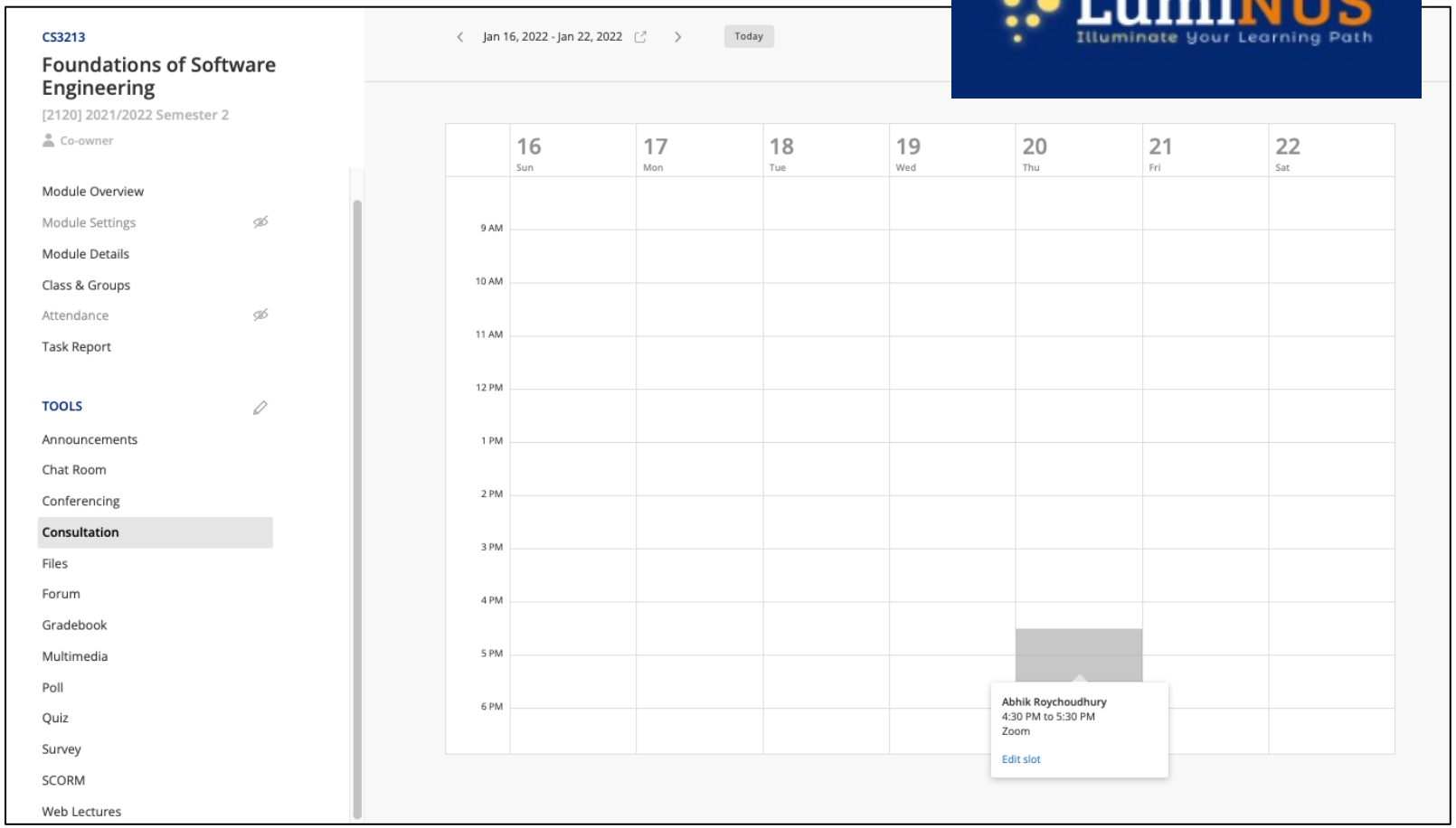
Contacts and Consultation

- General and assignment specific questions should be always submitted via the **LumiNUS forum**. This way you will get the fastest answer and all students can benefit.
- For question to your tutorial group you can ask your **tutor**.
- Otherwise, feel free to contact **Yannic** with all other questions concerning the lab-part of the course.
- Consider consultation slots we offer every week (next slide).

Contacts and Consultation

Week	Consultations
1	*- (no consultations)*
2	Lab
3	Lecture
4	Lab
5	Lecture
6	Lab (before long coding period)
R	*Recess Week (no consultations)*
7	Lecture (before midterm)
8	Lab
9	Lecture
10	Lab
11	Lecture
12	Lab (last lab in this week)
13	*- (no consultations)*
RD	Lecture
E	*Examination (no consultations)*

Thursday, 4:30 pm – 6 pm



Course Grading

Assessment

Project : 45%

Final Assessment: 40%

Midterm assessment: 15% (to be held online but during class)

https://luminus.nus.edu.sg/modules/76980563-4eb1-48cd-bd13-f3456650b0f1/details/module-description

Lab Grading consists of grading for:	Marks:
6 extra Assignment Sheets (A1, A2, A3, A4, A5, A7)	12
Intermediate Submission of project (A6)	6
Final Report and Presentation (A8+A9)	12
Final Code for Project (A8)	15
Total:	**45**

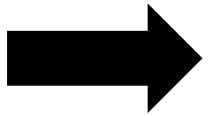
Aspects for the code submission: *correctness*, *completeness*, *maintainability*

Blue Slide

Do you have any questions so far about the course logistics?

Requirements

"The hardest single part of building a software system is **deciding** precisely **what** to build. No other part of the conceptual work is as difficult as establishing the detailed **technical requirements** ... No other part of the work so cripples the resulting system if done wrong. **No other part is as difficult to rectify later**."



Requirements Analysis & Elicitation

Brooks, F. P., "No silver bullet – essence and accidents of software engineering" in IEEE Computer, Vol. 20 (4), 10-19, 1987.

Requirements Elicitation Week 2

- You can experience requirement elicitation in the 3rd hour of the lecture in week 2 will be an interview session with two real stakeholders:
  - **Wei Tsang Ooi** (Associate Professor):
    CS1010 - Programming Methodology
  - **Alan, Cheng Ho-lun** (Senior Lecturer):
    CS 1102S - Data Structures and Algorithms
- Yannic will act as **moderator**

What needs to be done?

- ❏ **“as-is”** analysis
- ❏ **”to-be”** analysis

- ❏ **Terminology**
  → create a dictionary for special terms?
- ❏ **Documentation**: Traceability of Requirements?

“as-is” analysis

- Identification of the existing state:
  - What is the current state?
  - Why? How did the current state emerge?
  - What is good about the current state?
  - What is bad about the current state?

What is the benefit of the ”as-is” state?

It is “real”.

“as-is” analysis – challenges

- There are **implicit expectations** by the customer for the new system: everything which was acceptable or good in the current system, should remain unchanged or be improved.

> Software developers often do no realize that the customer does not primarly wants a **change** but an **improvement**.
> → negligence of the as-is analysis

- thankless task
- example: conversation about as-is state for door closing mechanism

You open the **door** at the **main entrance**?

Yes, this is what I told you.

Every **morning**?

Certainly.

Also at the **weekend**?

No, the entrance remains closed at the weekend.

Also at the **public holidays**?

No, obviously it remains closed.

What about if you are **sick** or on **leave**?

Then this is done by Mr. X.

And what about if Mr. X is not **available**?

Then at some point during the morning, a client will knock at the window because he or she cannot enter the store.

What means **morning**? …

“to-be” analysis

- Collection of **new** requirements
- Problems?
  - **unrealistic** expectations by client
  - **formulation/communication** by customers is incomprehensible by software people
  - there are multiple stakeholders representing the customer, but **not** every stakeholder has the **same expectations**
  - the customer/client is **internally** not perfectly organized
  - the client has desires and goals but no precise requirements
  - **unspoken/assumed** requirements by clients

What kind of analysis techniques do you know? And what do they target?

Requirement Analysis Techniques

	Main Focus		
Analysis Technique	“As-Is” State	“To-Be” State	Innovation Impact

Analysis of existent data and documents

Observation

Survey with { closed / structured / open } questions

Interview

Modelling

Experiments

Prototyping

Participative Development (wrt analysis)

How to find good questions?

- Which topics need to be covered in the requirement specification?
  - Purpose of the software
  - Functional Requirements
  - Requirements to External Interface
  - Requirements Regarding Technical Data
  - General Constraints and Requirements
  - Product Quality Requirements

1. Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Product overview
  - 1.3.1 Product perspective
  - 1.3.2 Product functions
  - 1.3.3 User characteristics
  - 1.3.4 Limitations
- 1.4 Definitions

2. References

3. Requirements
- 3.1 Functions
- 3.2 Performance requirements
- 3.3 Usability requirements
- 3.4 Interface requirements
- 3.5 Logical database requirements
- 3.6 Design constraints
- 3.7 Software system attributes
- 3.8 Supporting information

4. **Verification**
(parallel to subsections in Section 3)

5. Appendices
- 5.1 Assumptions and dependencies
- 5.2 Acronyms and abbreviations

SRS outline (IEEE 29148:2018)

ISO/IEC/IEEE 29148:2018, ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering, 2018, DOI: 10.1109/IEEESTD.2018.8559686 → page 67 to 74

Goal Oriented Requirement Engineering (GORE) Modeling

- Goal-oriented analysis focuses on the description and evaluation of alternatives and their relationship to the organizational objectives.
- Goals are introduced for pragmatic or engineering reasons – they help accomplish the objectives of several specific subtasks of RE
- Definition goal: A goal is a desirable state lying in the future, which is not reached automatically but by specific actions.
  - a Goal should be proactive.
- Goals and their dependencies are often described in conceptual models that are based on modeling languages.
- Definition goal model: “A goal model is a conceptual model. Its goals and decompositions are documented in sub-goals and as necessary further dependencies between (sub)-goals.“

Representations of Goals

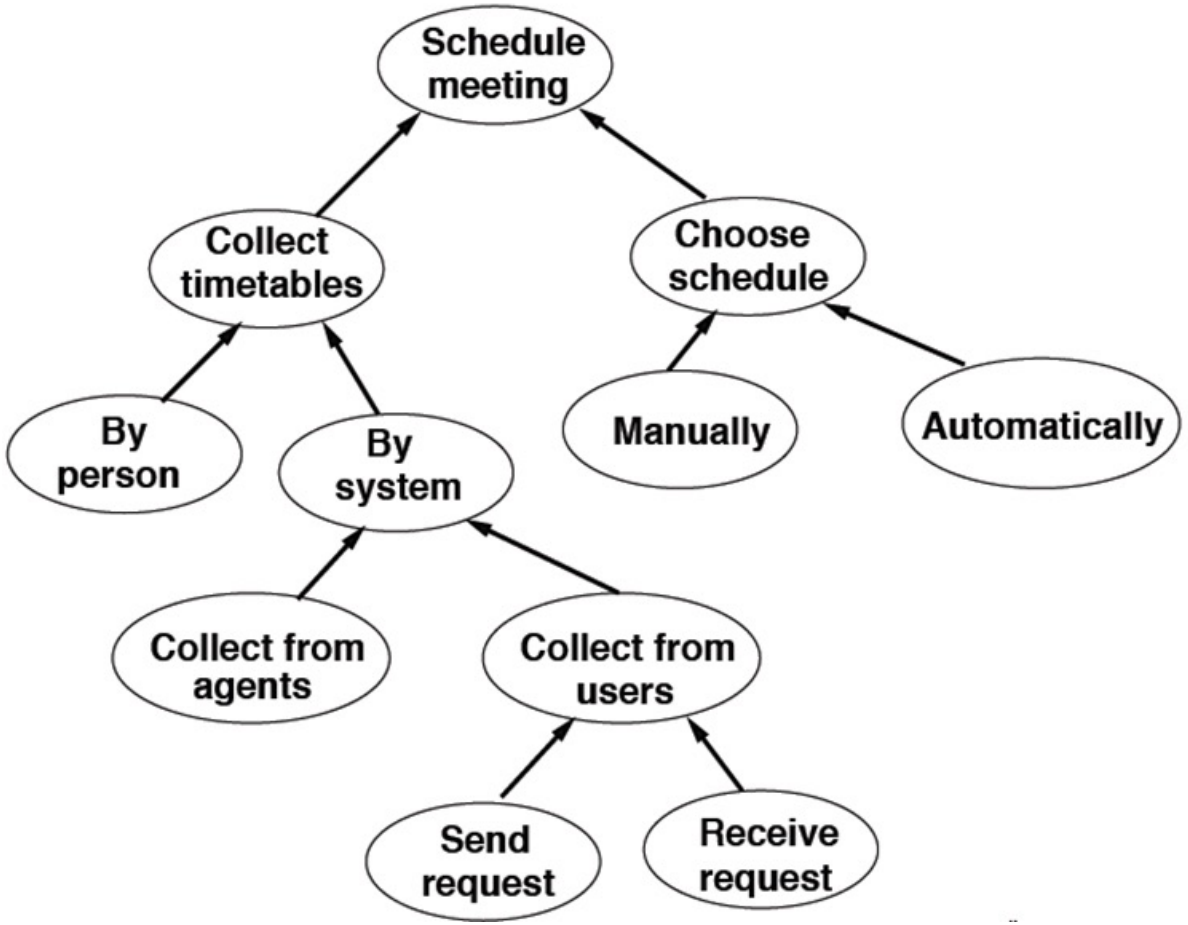
- **And / OR – Trees**
- **SIG (Softgoal interdependency graphs)**
- i-star (i*)
- GRL (Goal-oriented Requirement Language)

Example

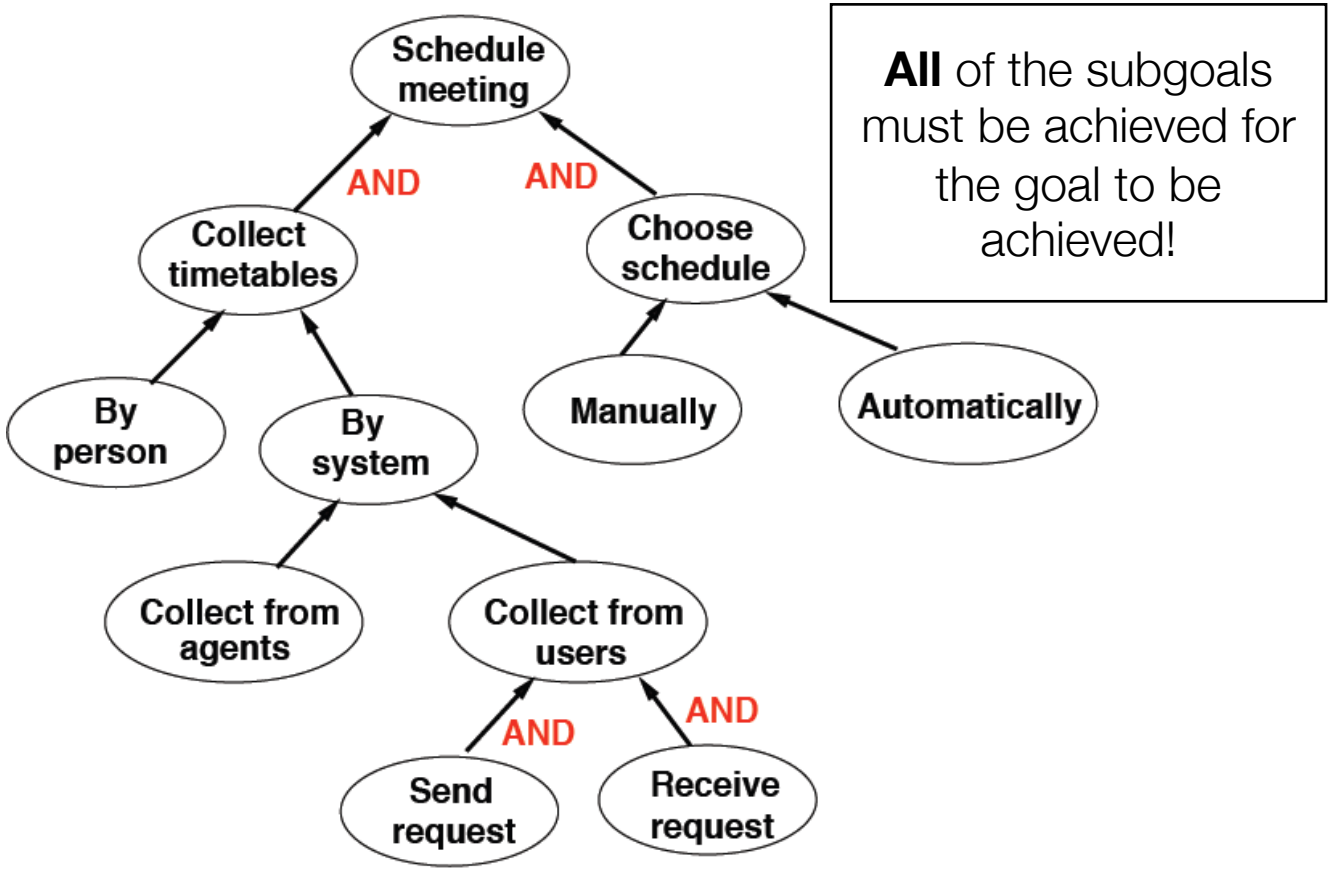
- Meeting Scheduler
  - Assists the initiator in scheduling a meeting
  - Meeting should be convenient for participants
  - Participants should be available

Based on slides by Betty H.C. Cheng, Professor, Computer Science and Engineering, Michigan State University

Goal Model

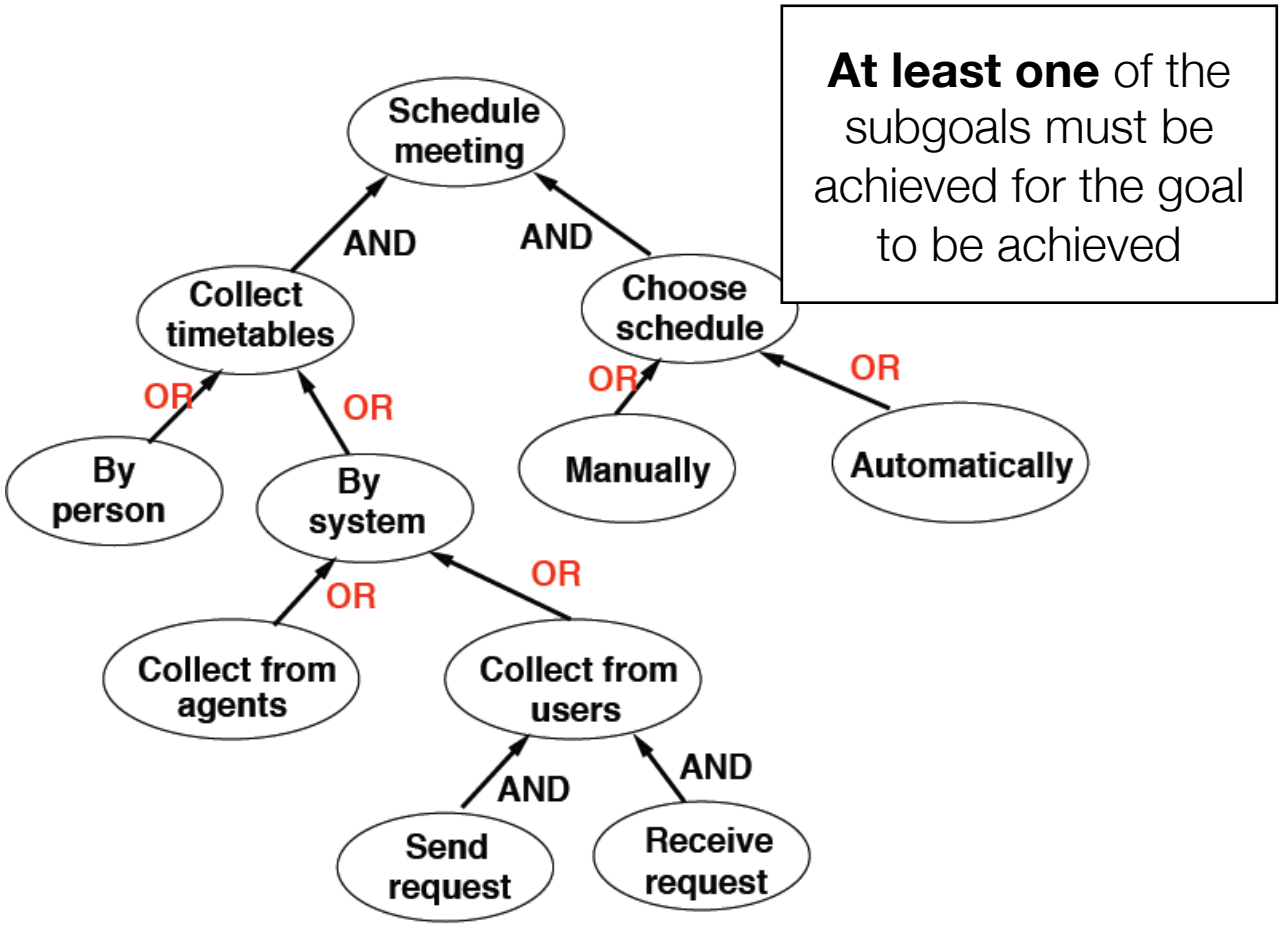

Schedule meeting
Collect timetables
Choose schedule
By person
By system
Manually
Automatically
Collect from agents
Collect from users
Send request
Receive request

AND-Refinement Tree



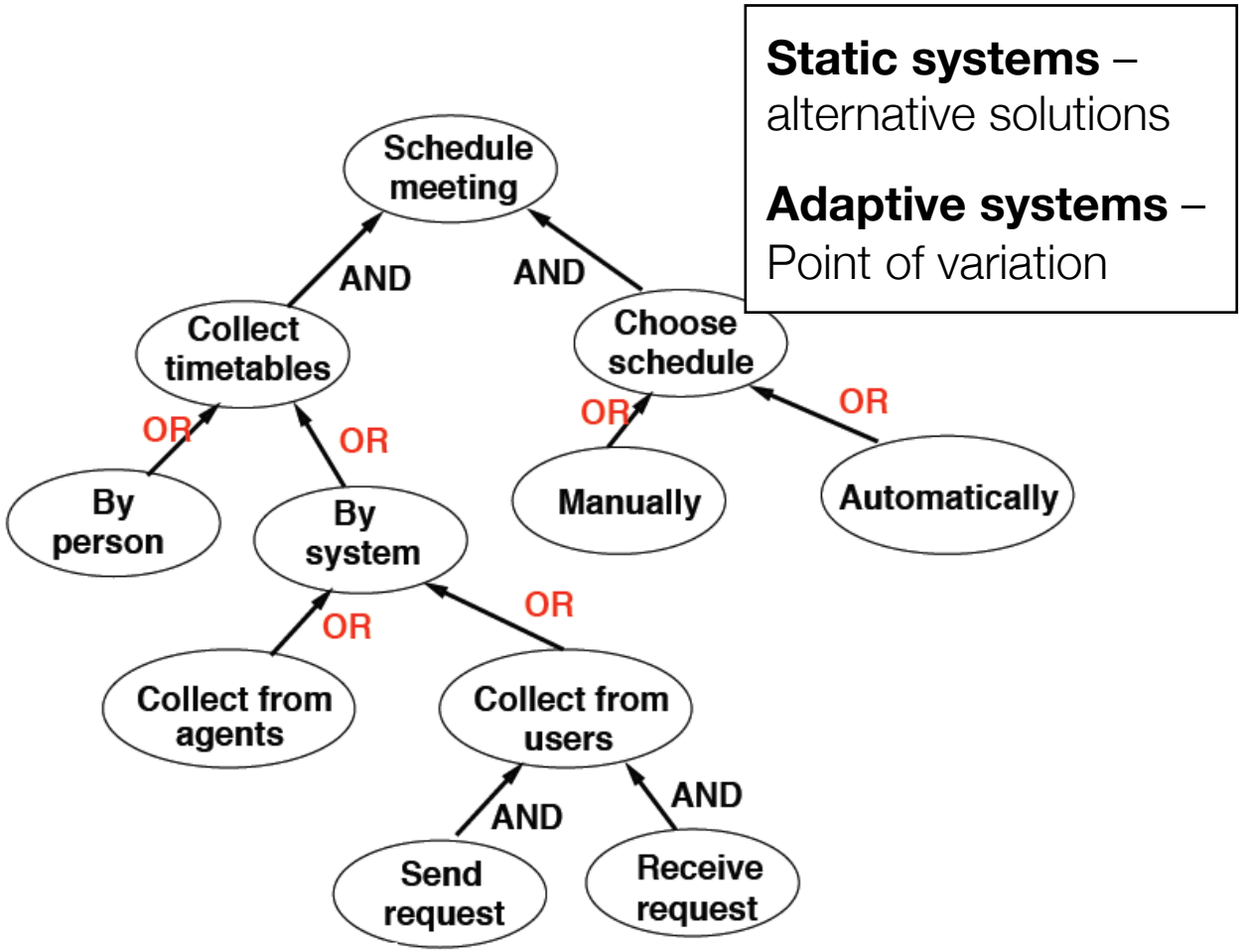
All of the subgoals must be achieved for the goal to be achieved!

OR-Refinement Tree



At least one of the subgoals must be achieved for the goal to be achieved

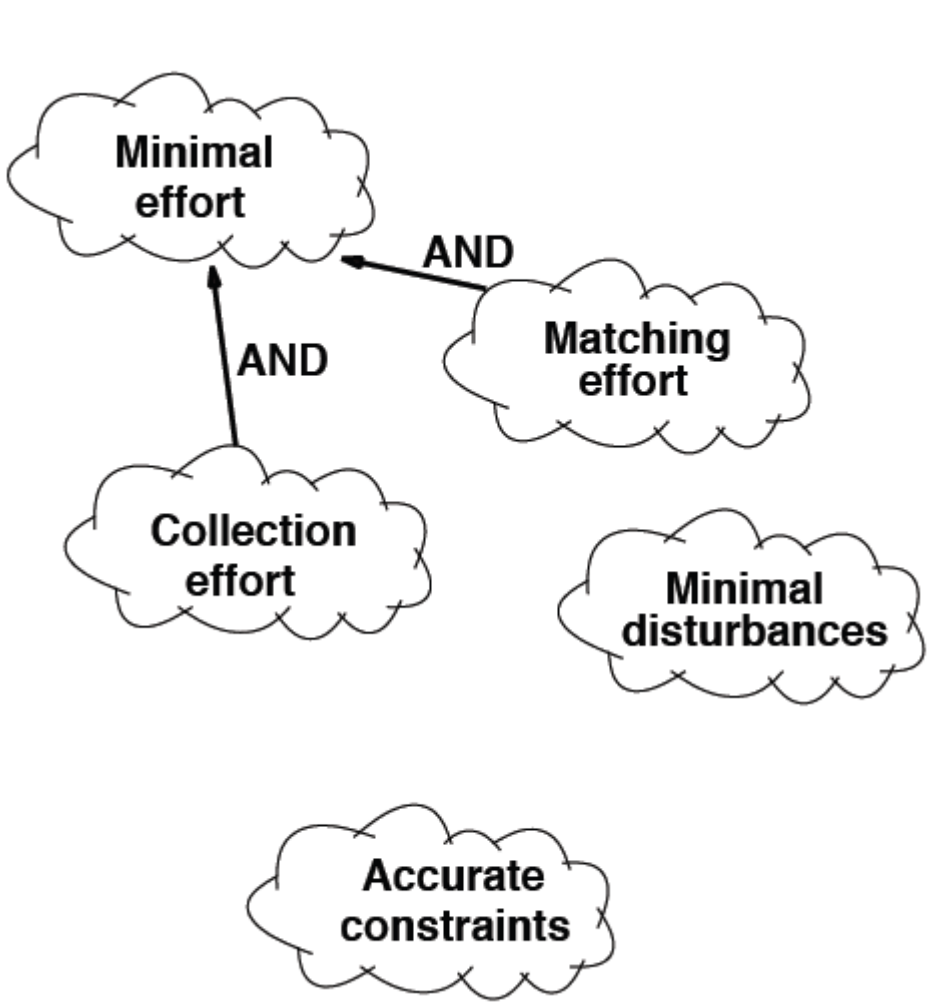
Interpretations of OR Refinements

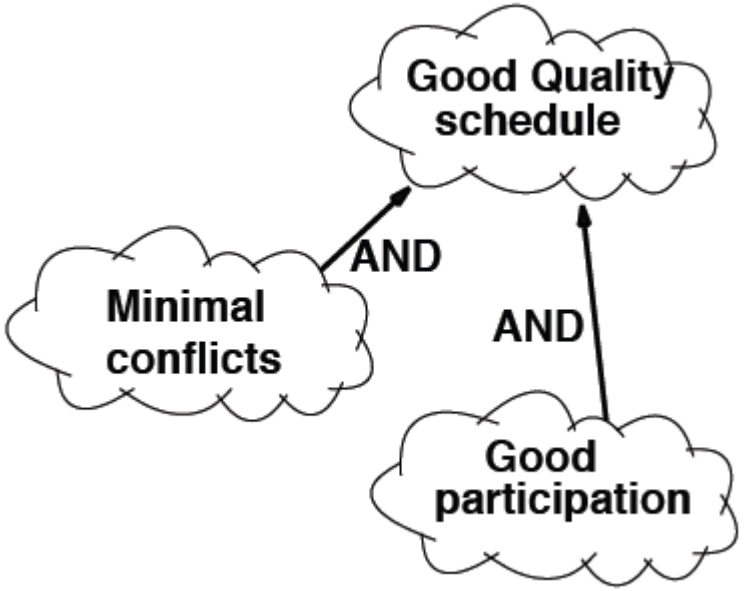


Static systems – alternative solutions

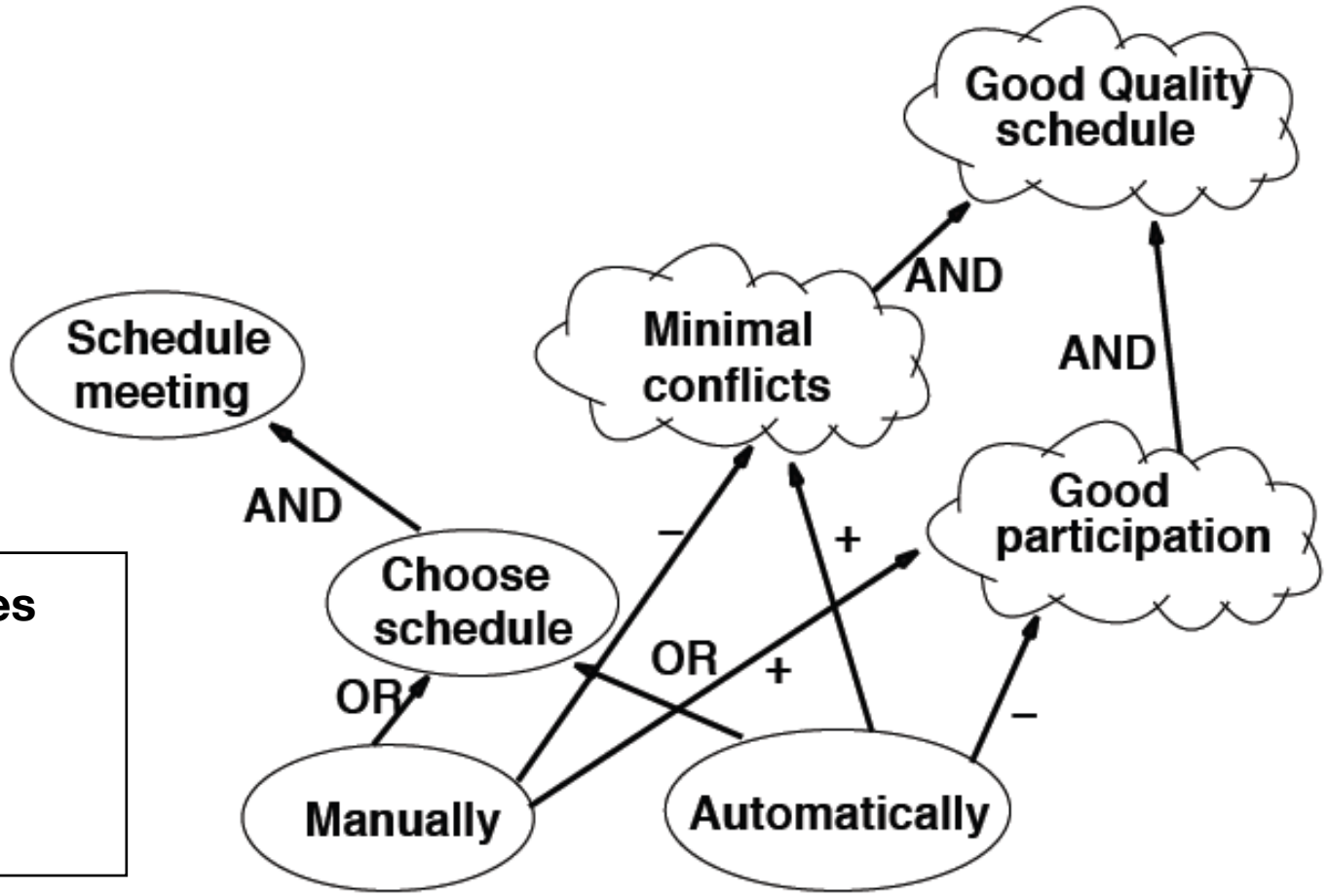
Adaptive systems – Point of variation

Soft Goal Graphs





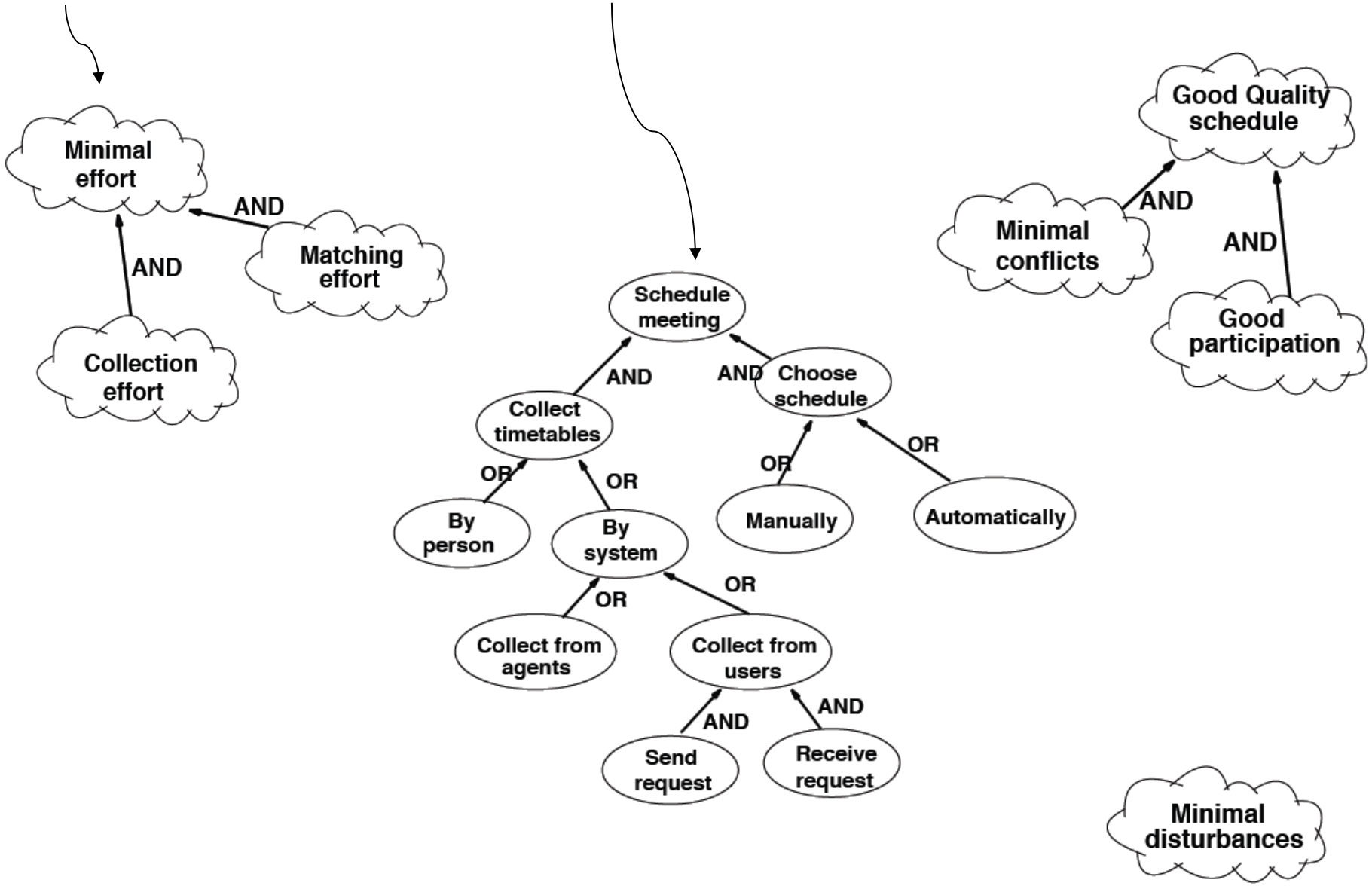
Contributions to Softgoals



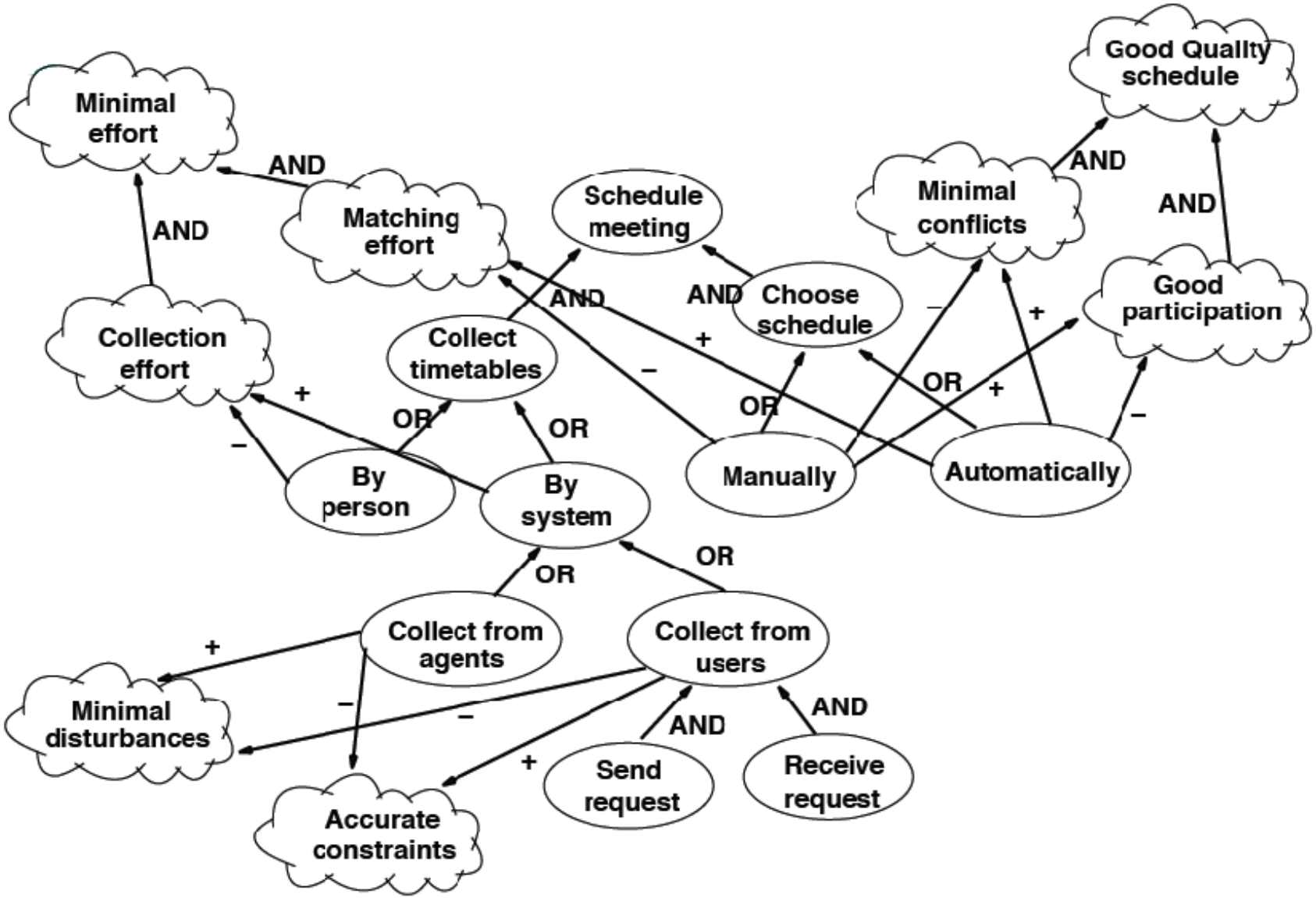
Evaluate alternatives

- ❑ Helps (+)
- ❑ Makes (++)
- ❑ Hurts (-)
- ❑ Breaks (--)

Soft Goals & Hard Goals



How to combine



Any questions about Goals?

Conclusion

- **Lab** is the practical extension of the lecture. Key aspect will be the **course project**.

Deadline for group registration in Google Sheet:
Friday, Jan 21, 10 am

Next Week:
Requirement Elicitation Session

- Focus on Assignment 1 and **prepare** for the session!
- **Everyone** is expected to **ask questions** and **interact** with the customer.
- Keep in mind that you will have to model the requirements.