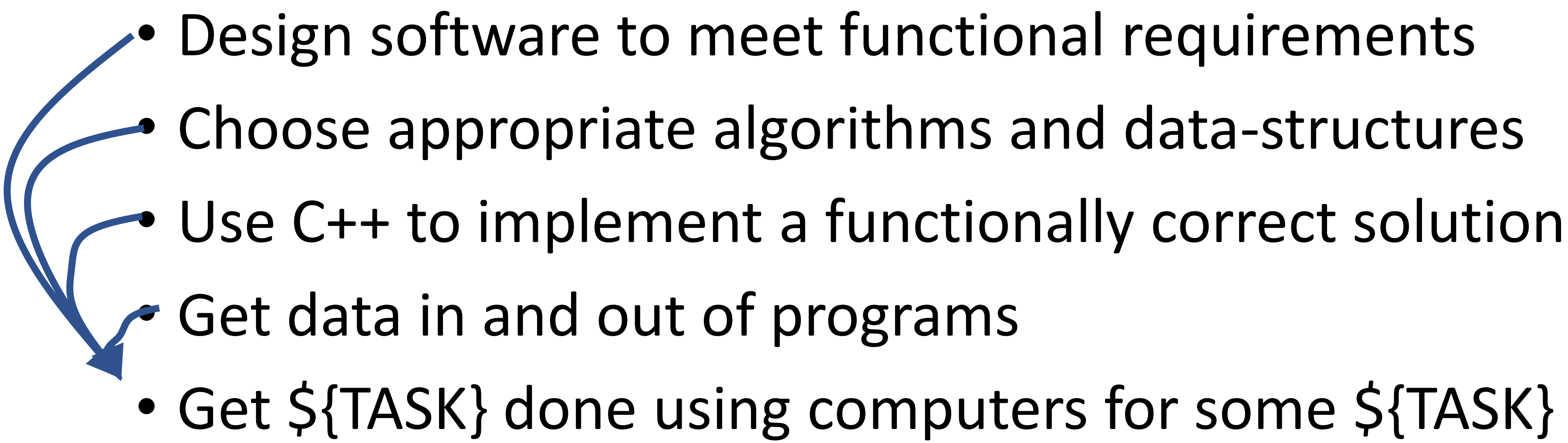
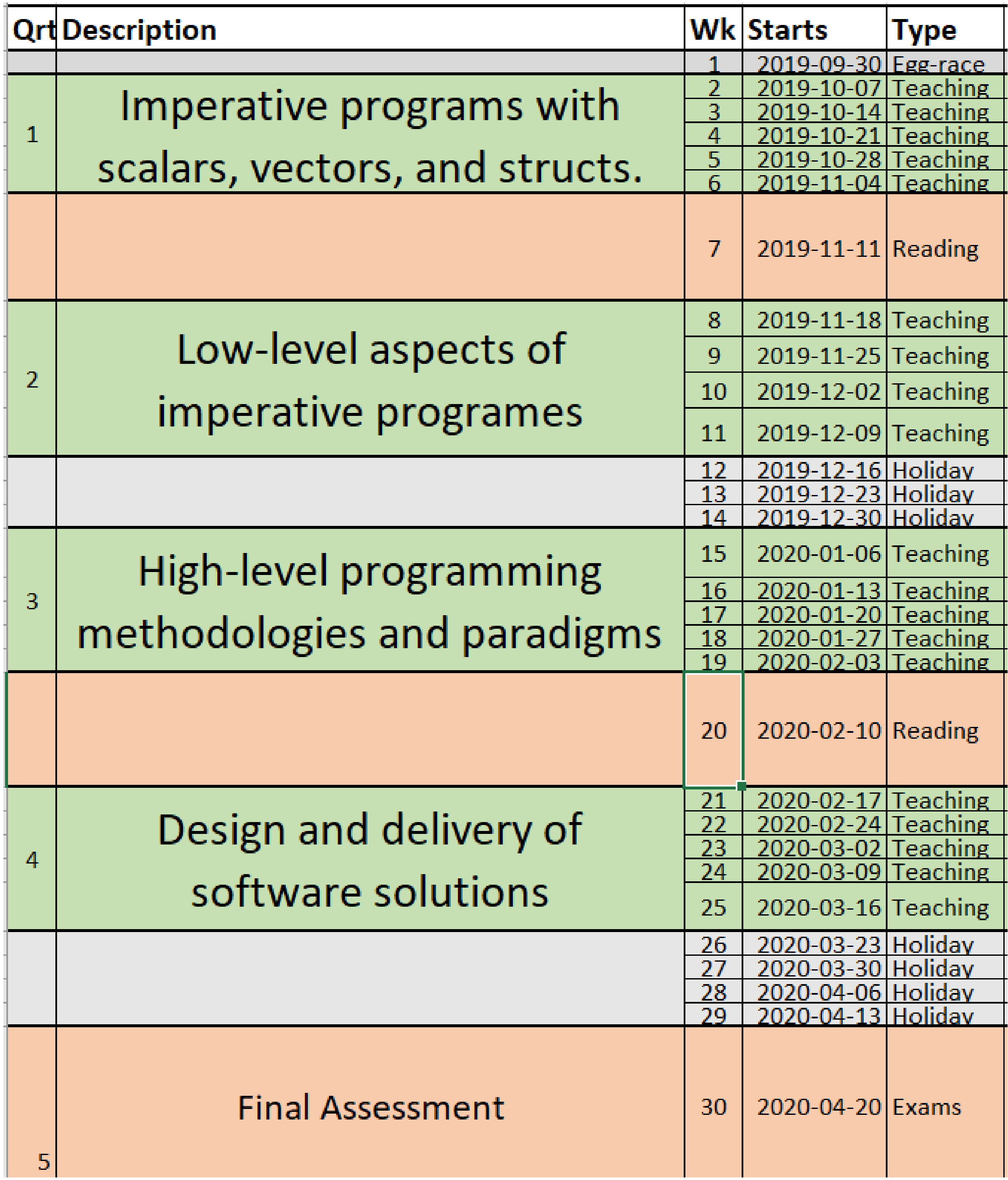
Assessment

PROGRAMMING FOR ENGINEERS

By the end of this course you should be able to:

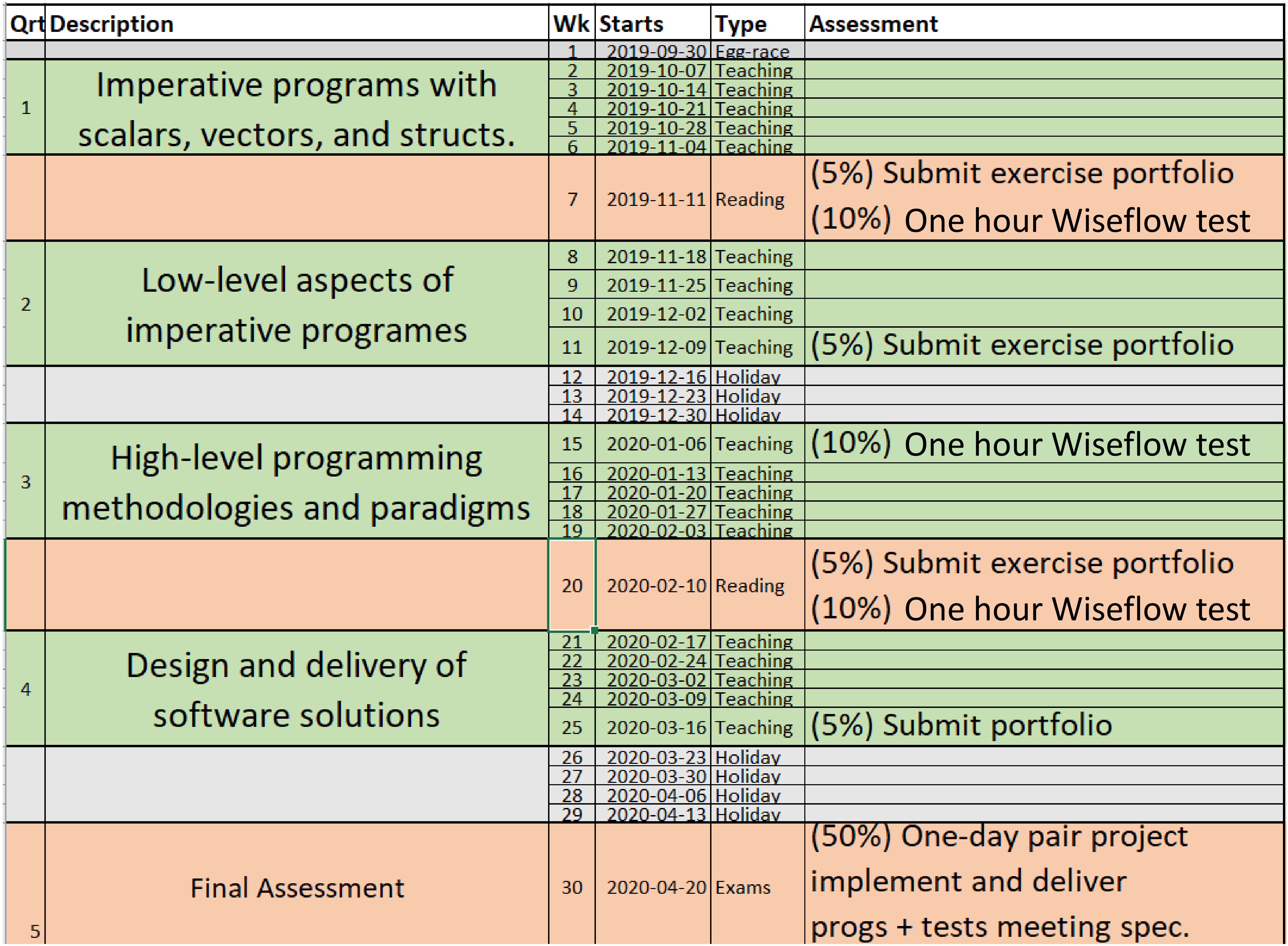


- Design software to meet functional requirements
- Choose appropriate algorithms and data-structures
- Use C++ to implement a functionally correct solution
- Get data in and out of programs
- Get ${TASK} done using computers for some ${TASK}



Qrt	Description	Wk	Starts	Type
		1	2019-09-30	Egg-race
1	Imperative programs with scalars, vectors, and structs.	2	2019-10-07	Teaching
		3	2019-10-14	Teaching
		4	2019-10-21	Teaching
		5	2019-10-28	Teaching
		6	2019-11-04	Teaching
		7	2019-11-11	Reading
2	Low-level aspects of imperative programes	8	2019-11-18	Teaching
		9	2019-11-25	Teaching
		10	2019-12-02	Teaching
		11	2019-12-09	Teaching
		12	2019-12-16	Holiday
		13	2019-12-23	Holiday
		14	2019-12-30	Holiday
3	High-level programming methodologies and paradigms	15	2020-01-06	Teaching
		16	2020-01-13	Teaching
		17	2020-01-20	Teaching
		18	2020-01-27	Teaching
		19	2020-02-03	Teaching
		20	2020-02-10	Reading
4	Design and delivery of software solutions	21	2020-02-17	Teaching
		22	2020-02-24	Teaching
		23	2020-03-02	Teaching
		24	2020-03-09	Teaching
		25	2020-03-16	Teaching
		26	2020-03-23	Holiday
		27	2020-03-30	Holiday
		28	2020-04-06	Holiday
		29	2020-04-13	Holiday
5	Final Assessment	30	2020-04-20	Exams

You are here



Qrt	Description	Wk	Starts	Type	Assessment
		1	2019-09-30	Egg-race	
1	Imperative programs with scalars, vectors, and structs.	2	2019-10-07	Teaching	
		3	2019-10-14	Teaching	
		4	2019-10-21	Teaching	
		5	2019-10-28	Teaching	
		6	2019-11-04	Teaching	
		7	2019-11-11	Reading	(5%) Submit exercise portfolio (10%) One hour Wiseflow test
2	Low-level aspects of imperative programes	8	2019-11-18	Teaching	
		9	2019-11-25	Teaching	
		10	2019-12-02	Teaching	
		11	2019-12-09	Teaching	(5%) Submit exercise portfolio
		12	2019-12-16	Holiday	
		13	2019-12-23	Holiday	
		14	2019-12-30	Holiday	
3	High-level programming methodologies and paradigms	15	2020-01-06	Teaching	(10%) One hour Wiseflow test
		16	2020-01-13	Teaching	
		17	2020-01-20	Teaching	
		18	2020-01-27	Teaching	
		19	2020-02-03	Teaching	
		20	2020-02-10	Reading	(5%) Submit exercise portfolio (10%) One hour Wiseflow test
4	Design and delivery of software solutions	21	2020-02-17	Teaching	
		22	2020-02-24	Teaching	
		23	2020-03-02	Teaching	
		24	2020-03-09	Teaching	
		25	2020-03-16	Teaching	(5%) Submit portfolio
		26	2020-03-23	Holiday	
		27	2020-03-30	Holiday	
		28	2020-04-06	Holiday	
		29	2020-04-13	Holiday	
5	Final Assessment	30	2020-04-20	Exams	(50%) One-day pair project implement and deliver progs + tests meeting spec.

Types of assessments

- Exercise portfolio (20%)
  - Simple exercises to ensure people keep up
  - Probably everyone gets close to 100%

- Mid-term/start-of-term assessments (30%)
  - Make sure you are taking onboard the knowledge

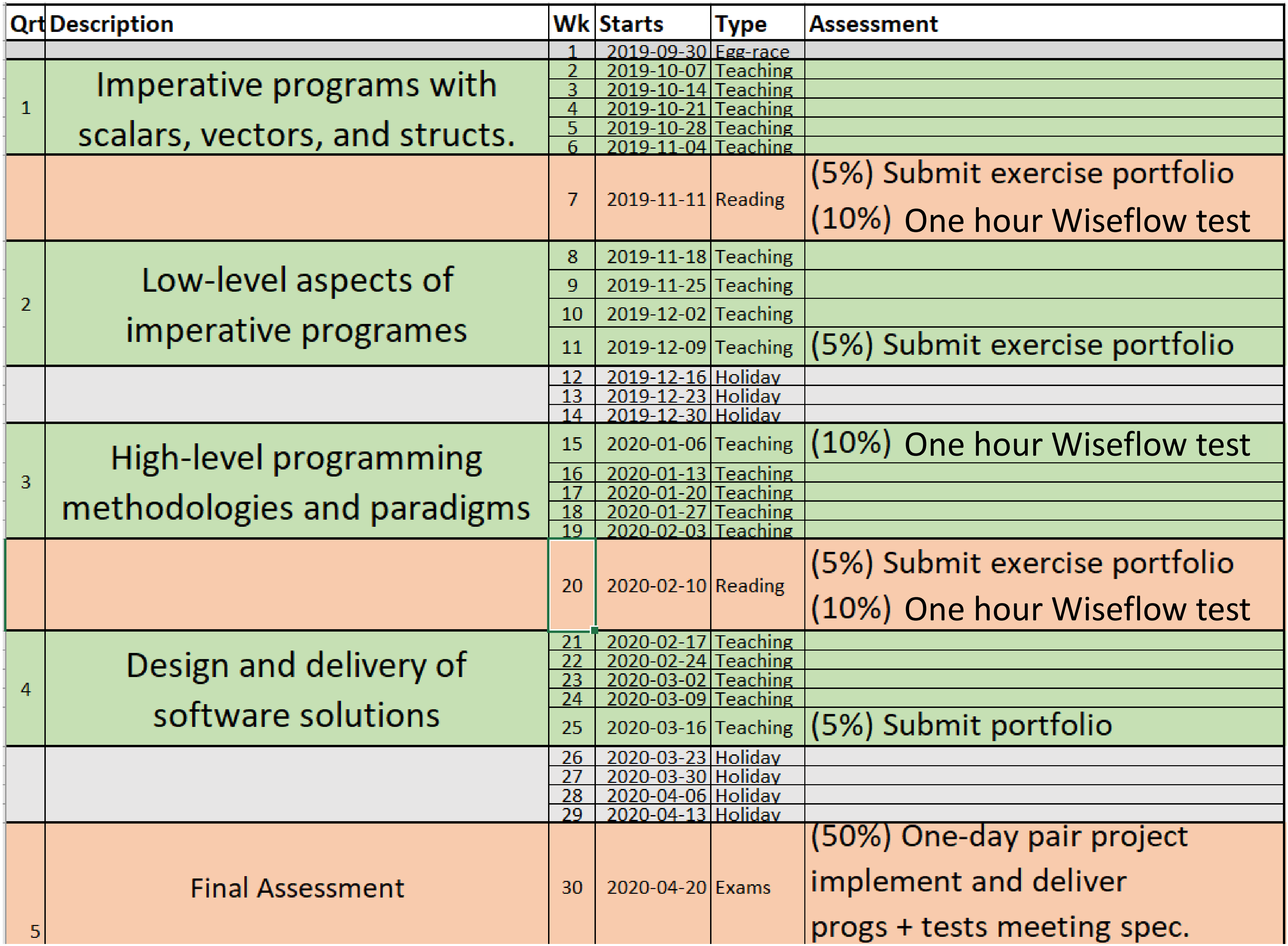
- Final assessment (50%)
  - Single one day piece of coursework
  - Can you create a working software solution?

Exact scheduling

- Post-Christmas test
  - Follows the format of mid-term:
    - Wiseflow, multiple choice, 12 questions, 50 minutes
    - Questions cover topics from this quarter
    - Similar level of difficulty/easiness to mid-term
  - Planned to be on morning of Monday 6th
    - Before the maths test on Wednesday 8th
    - Keeps all assessments before lectures start
- Final one day assessment
  - Likely to be Monday of the 2nd week
  - Just after exams week; just before project

Meta-learning

Why do we teach the way we teach?



Qrt	Description	Wk	Starts	Type	Assessment
		1	2019-09-30	Egg-race	
1	Imperative programs with scalars, vectors, and structs.	2	2019-10-07	Teaching	
		3	2019-10-14	Teaching	
		4	2019-10-21	Teaching	
		5	2019-10-28	Teaching	
		6	2019-11-04	Teaching	
		7	2019-11-11	Reading	(5%) Submit exercise portfolio (10%) One hour Wiseflow test
2	Low-level aspects of imperative programes	8	2019-11-18	Teaching	
		9	2019-11-25	Teaching	
		10	2019-12-02	Teaching	
		11	2019-12-09	Teaching	(5%) Submit exercise portfolio
		12	2019-12-16	Holiday	
		13	2019-12-23	Holiday	
		14	2019-12-30	Holiday	
3	High-level programming methodologies and paradigms	15	2020-01-06	Teaching	(10%) One hour Wiseflow test
		16	2020-01-13	Teaching	
		17	2020-01-20	Teaching	
		18	2020-01-27	Teaching	
		19	2020-02-03	Teaching	
		20	2020-02-10	Reading	(5%) Submit exercise portfolio (10%) One hour Wiseflow test
4	Design and delivery of software solutions	21	2020-02-17	Teaching	
		22	2020-02-24	Teaching	
		23	2020-03-02	Teaching	
		24	2020-03-09	Teaching	
		25	2020-03-16	Teaching	(5%) Submit portfolio
		26	2020-03-23	Holiday	
		27	2020-03-30	Holiday	
		28	2020-04-06	Holiday	
		29	2020-04-13	Holiday	
5	Final Assessment	30	2020-04-20	Exams	(50%) One-day pair project implement and deliver progs + tests meeting spec.

Why is everything so last minute?

- Everything gets done just in time
  - Lecture slides uploaded minutes before the lecture
  - Lab exercises uploaded just before the lab
  - Portfolio exercises being changed just before release
- Some things didn’t appear as expected
  - e.g. source control has not appeared much

Is the course being written as we go?

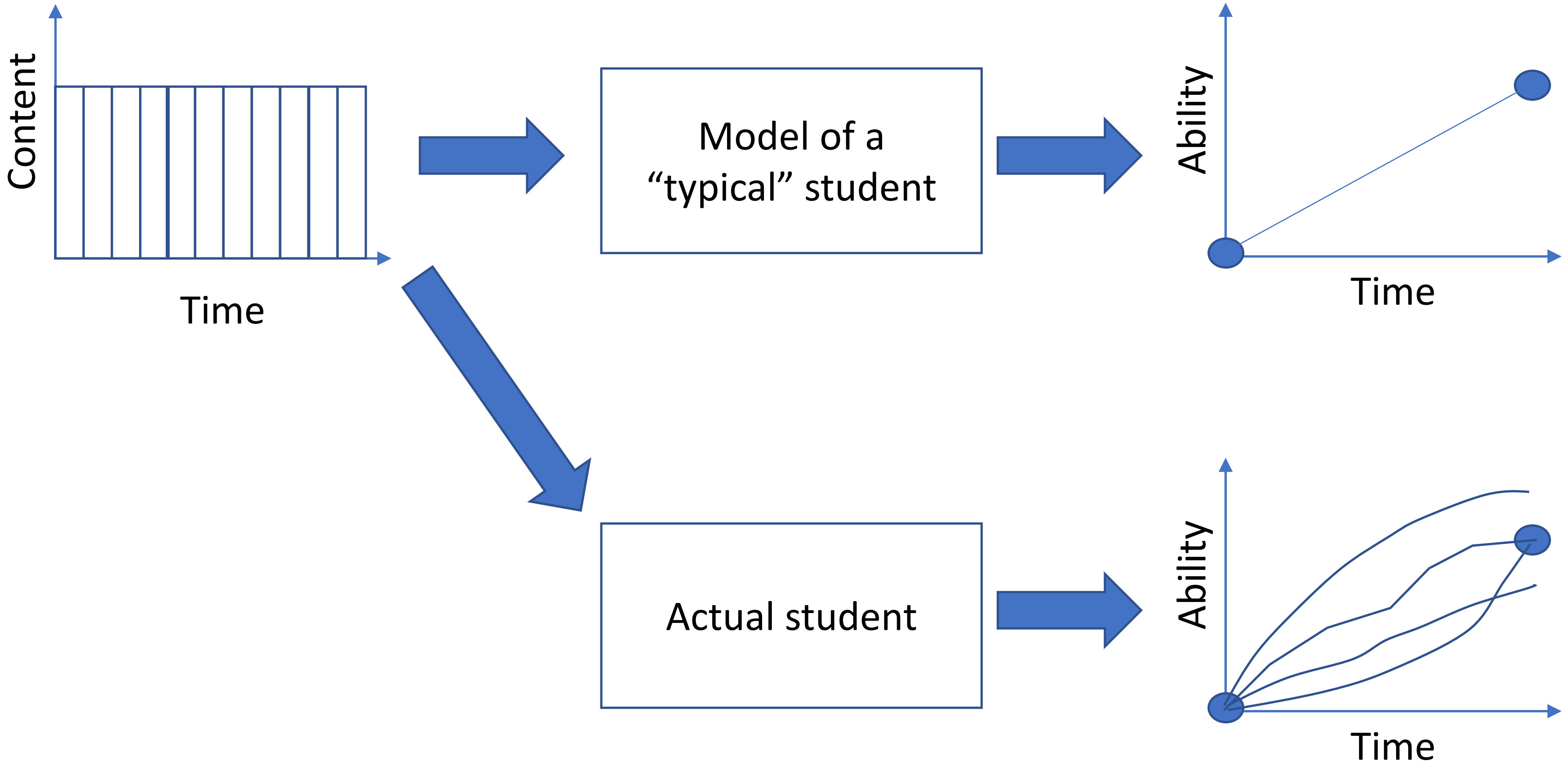
The course is being ***re-***written as we go

Open loop vs Closed loop

- We want to achieve certain objectives on this degree
  - Three years ago: objectives for the degree and course
  - Two years ago: objectives for each module
  - One year ago: detailed plan for each module
    - Content, structure, assessments, ...
  - Six months ago: full content for module
    - Lectures, exercises, labs, ...
- We have to guess how students will react
  - Learning is a complex subject
  - We might have designed it wrong
- Delivering exactly as planned would be ***open-loop***

Open-loop vs closed-loop is 2nd year control theory. You will ***not*** be assessed on it here...

Open-loop control



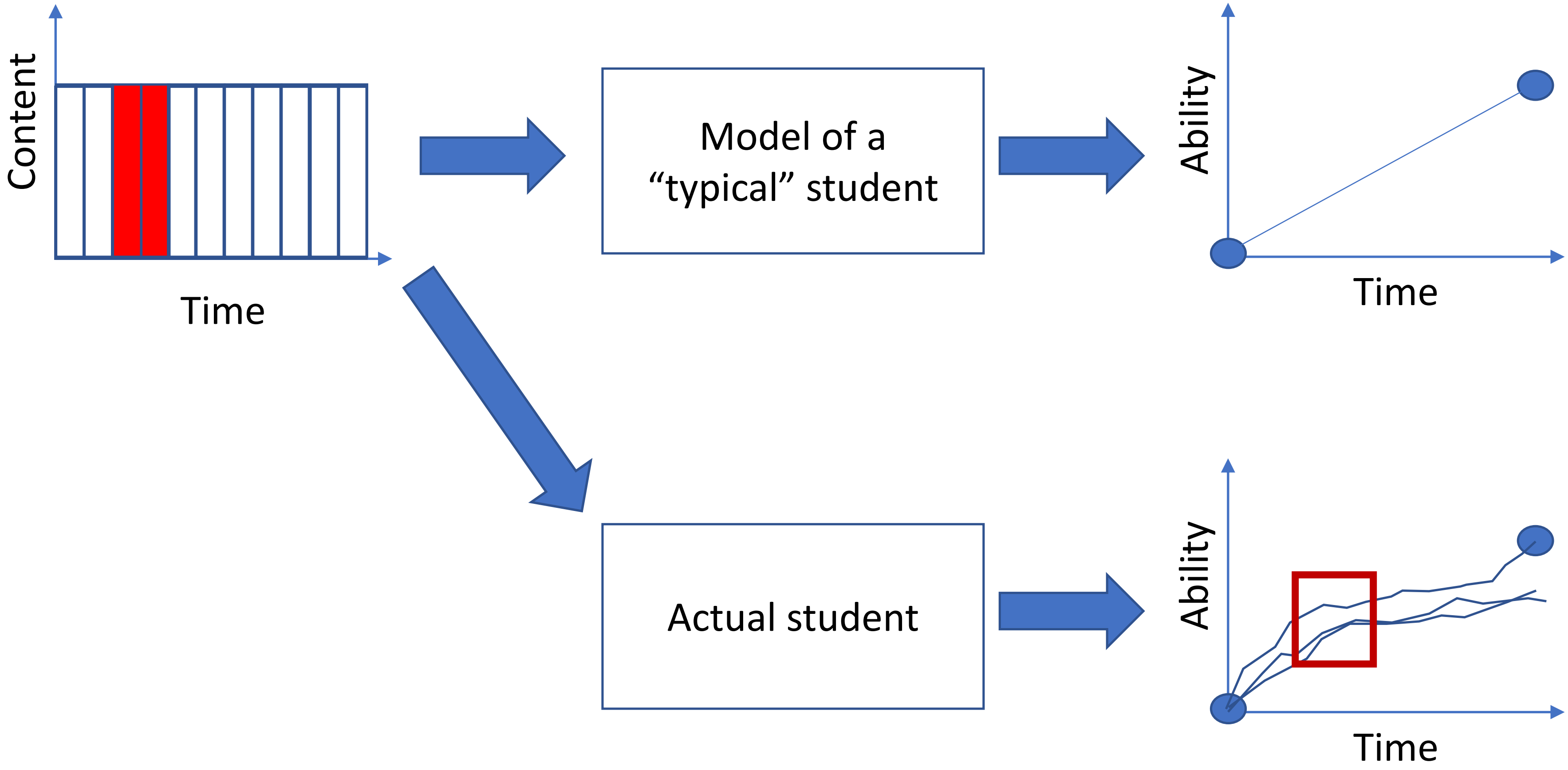
Model of a “typical” student

Actual student

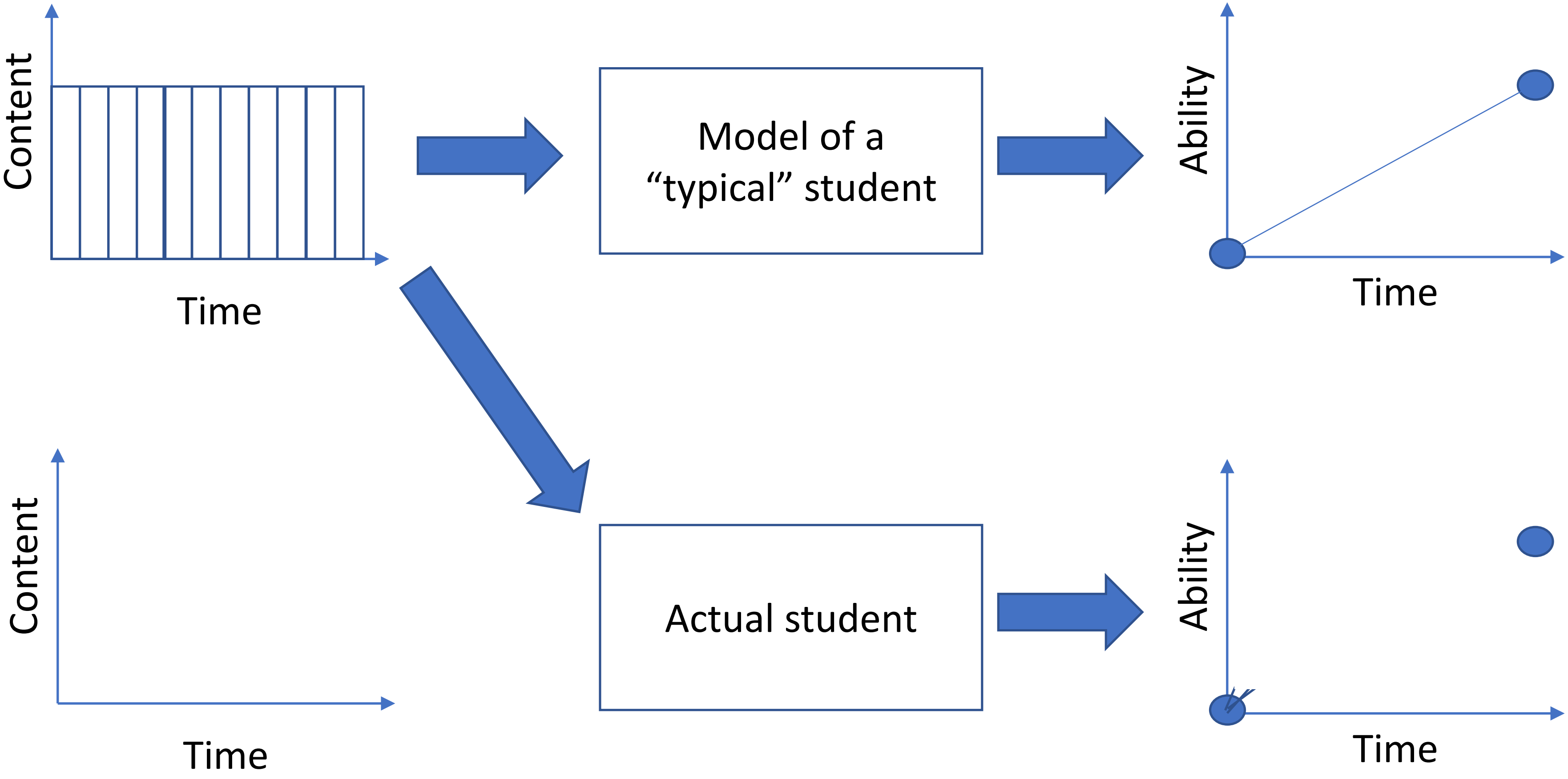
Pros and cons of open-loop

- *Positives*: open-loop is efficient
    - Can decide everything ahead of time
    - Can parallelise the implementation
    - Can schedule work in-between other classes
    - No dependencies to slow things down
- *Negatives*: open-loop needs a very good model
    - Need to understand the “transfer function”
        - How will students respond to input?
        - Does new teaching method actually increase learning rate?
        - How quickly can students internalize input in this order?
    - Must ensure there are no known-unknowns
        - Does assessment timing X actually make things worse or better

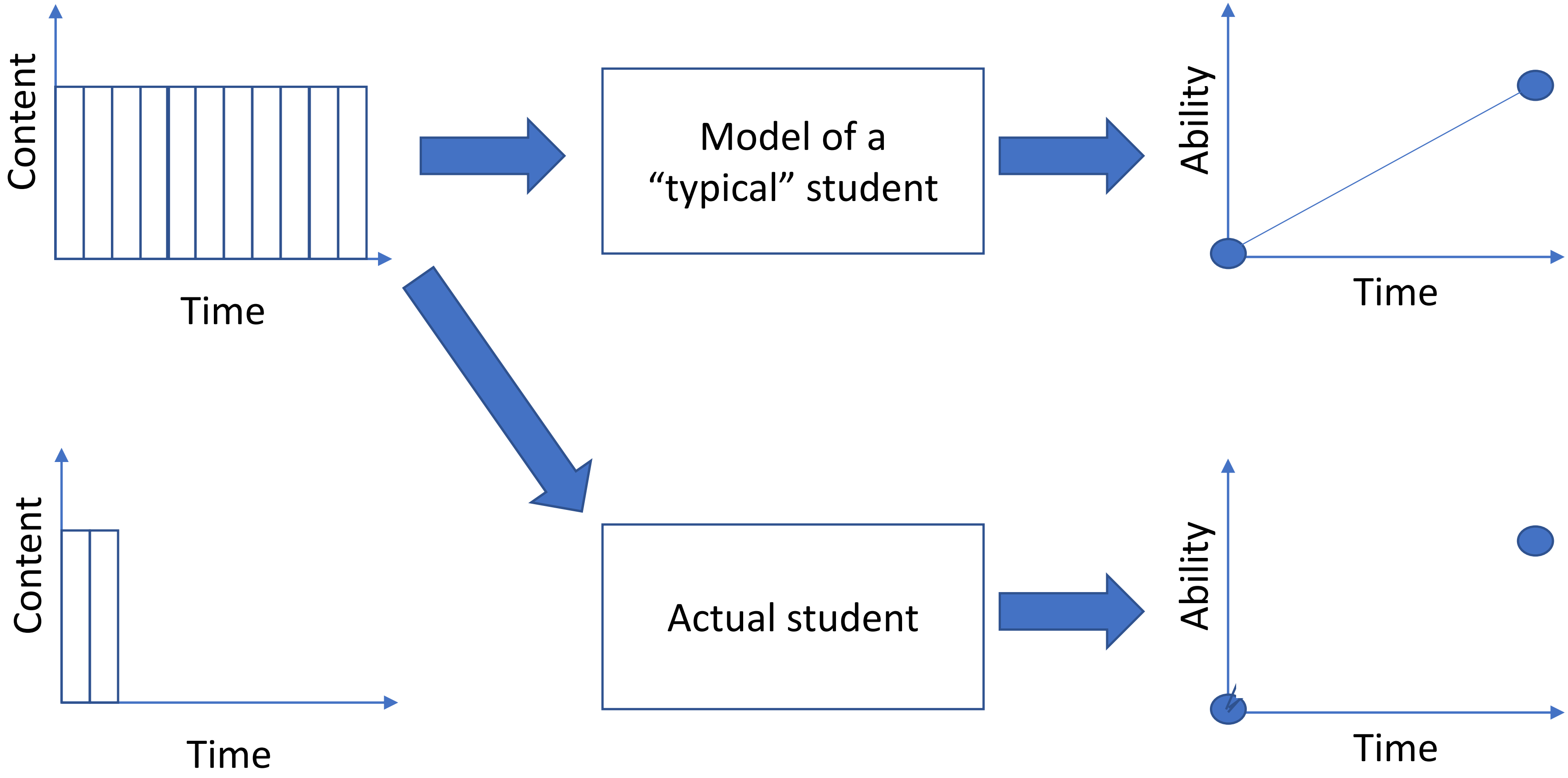
Open-loop control



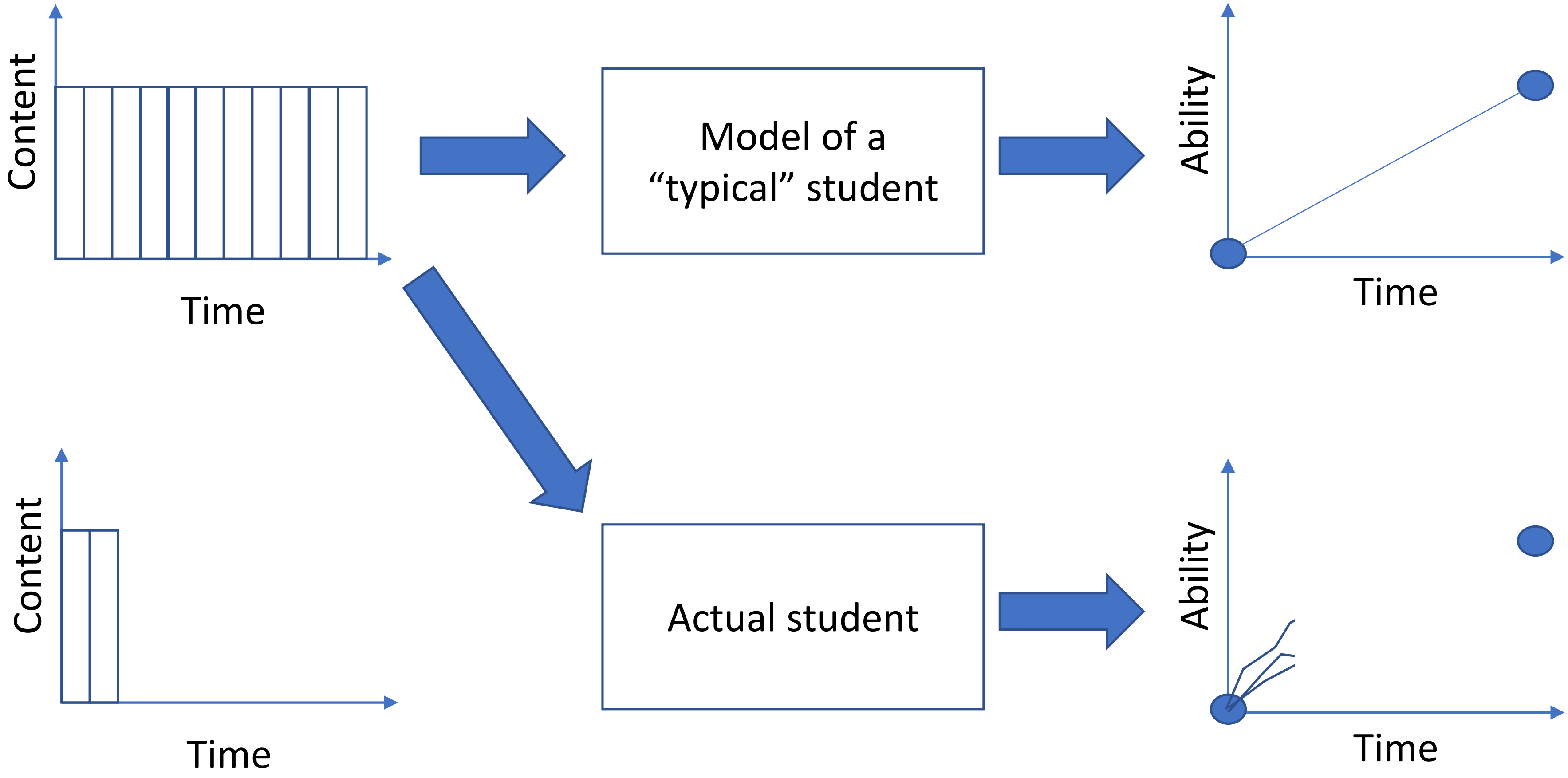
Closed-loop control



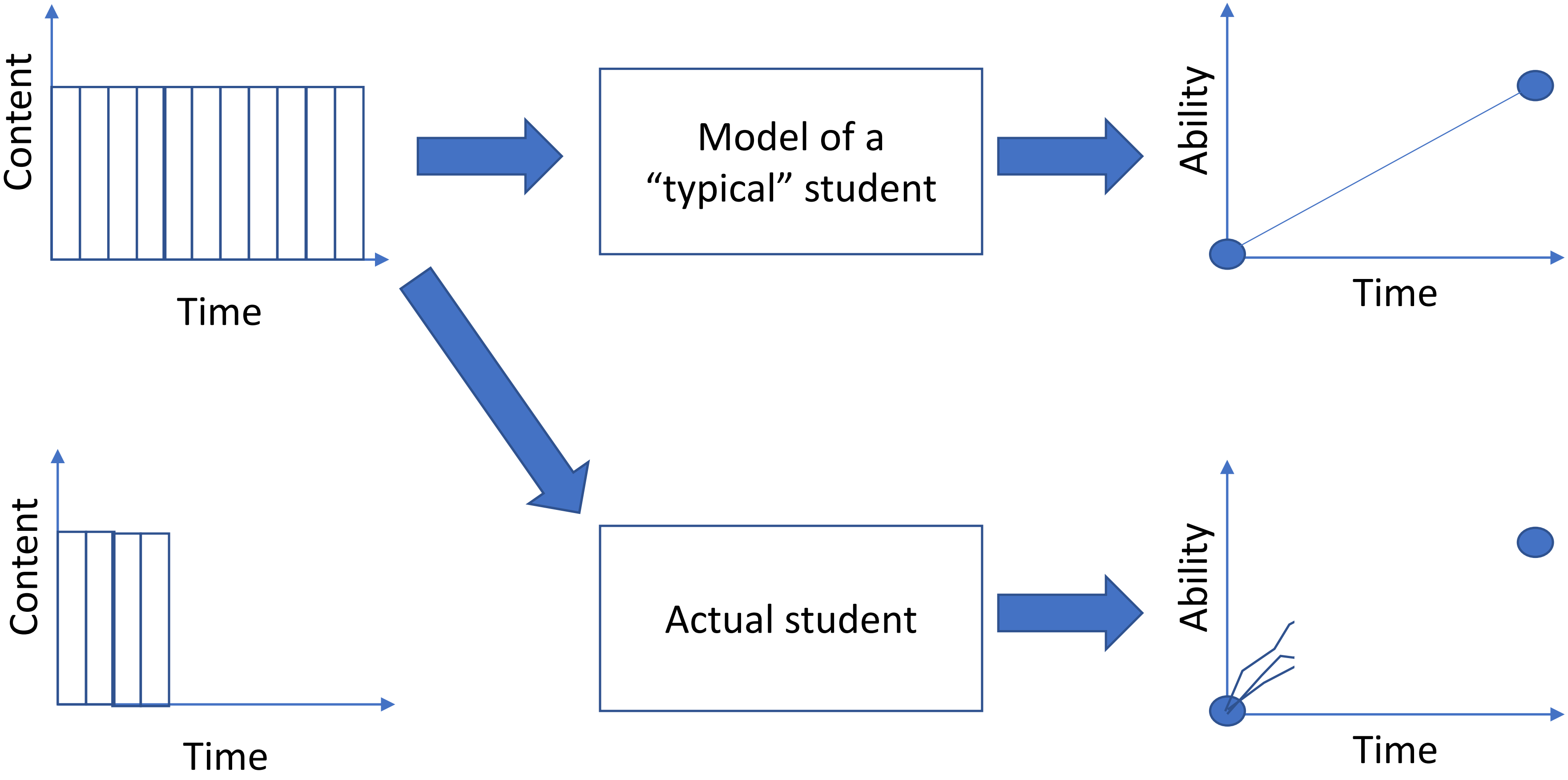
Closed-loop control



Closed-loop control



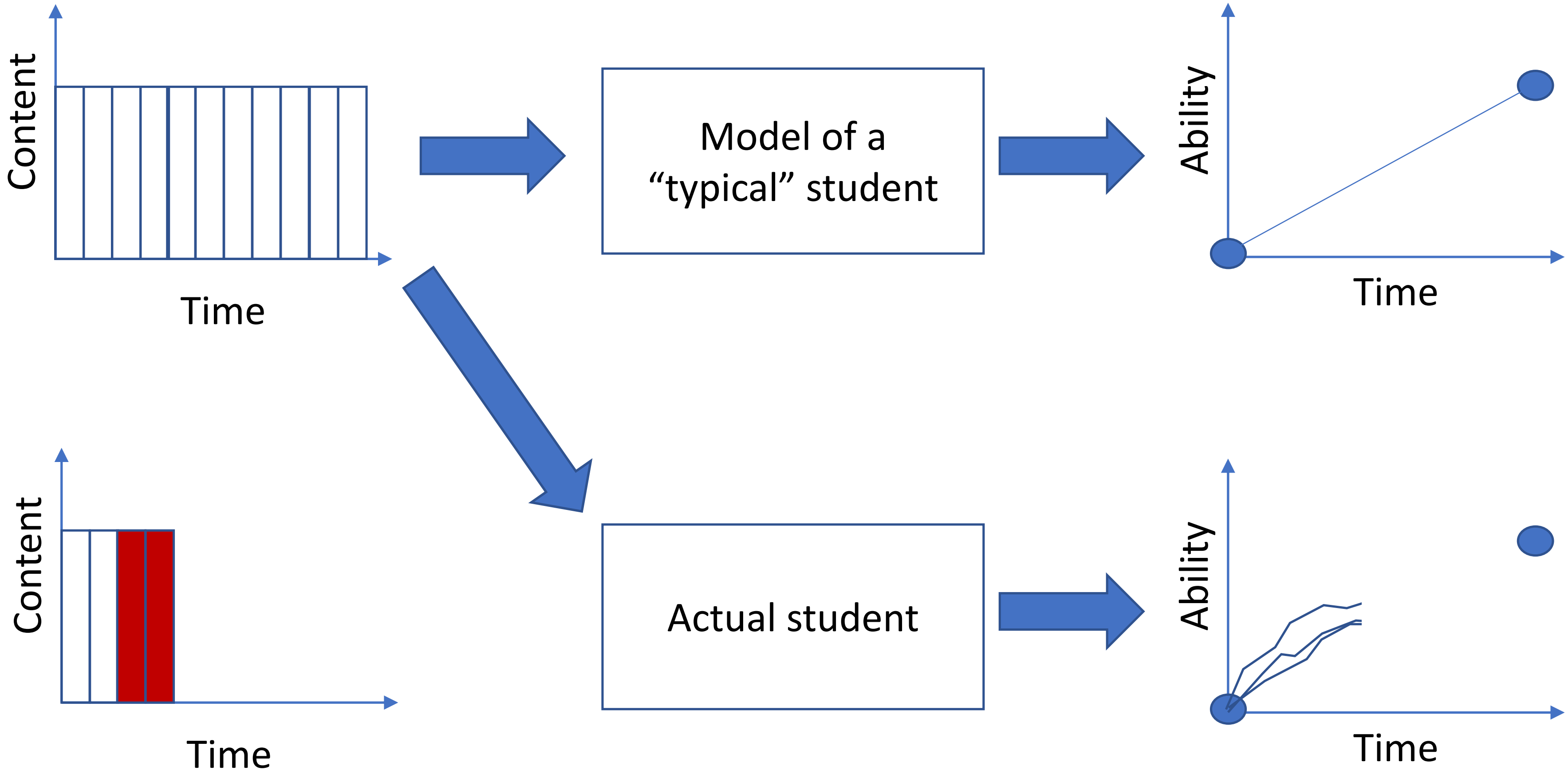
Closed-loop control



Model of a “typical” student

Actual student

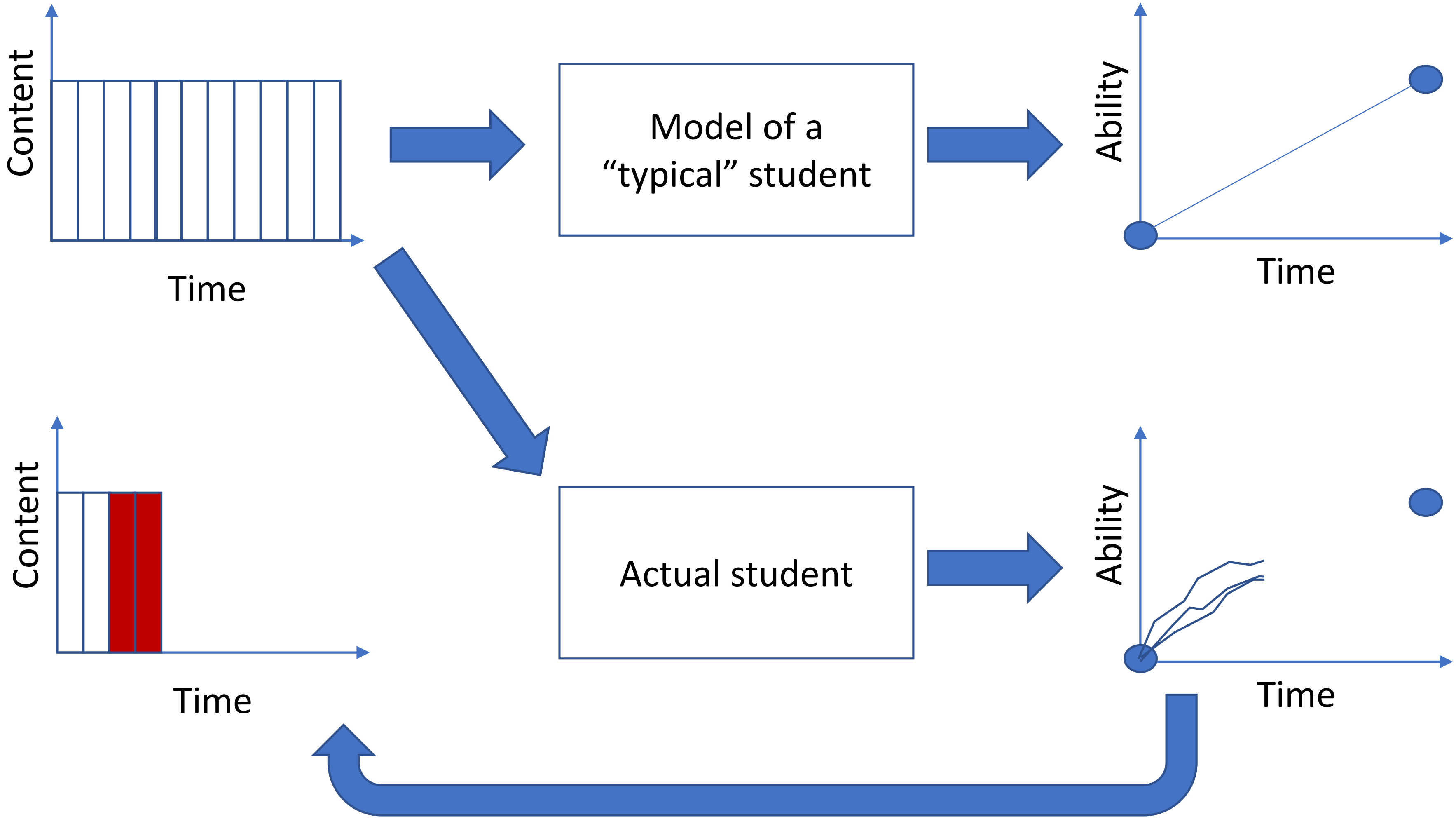
Closed-loop control



Model of a “typical” student

Actual student

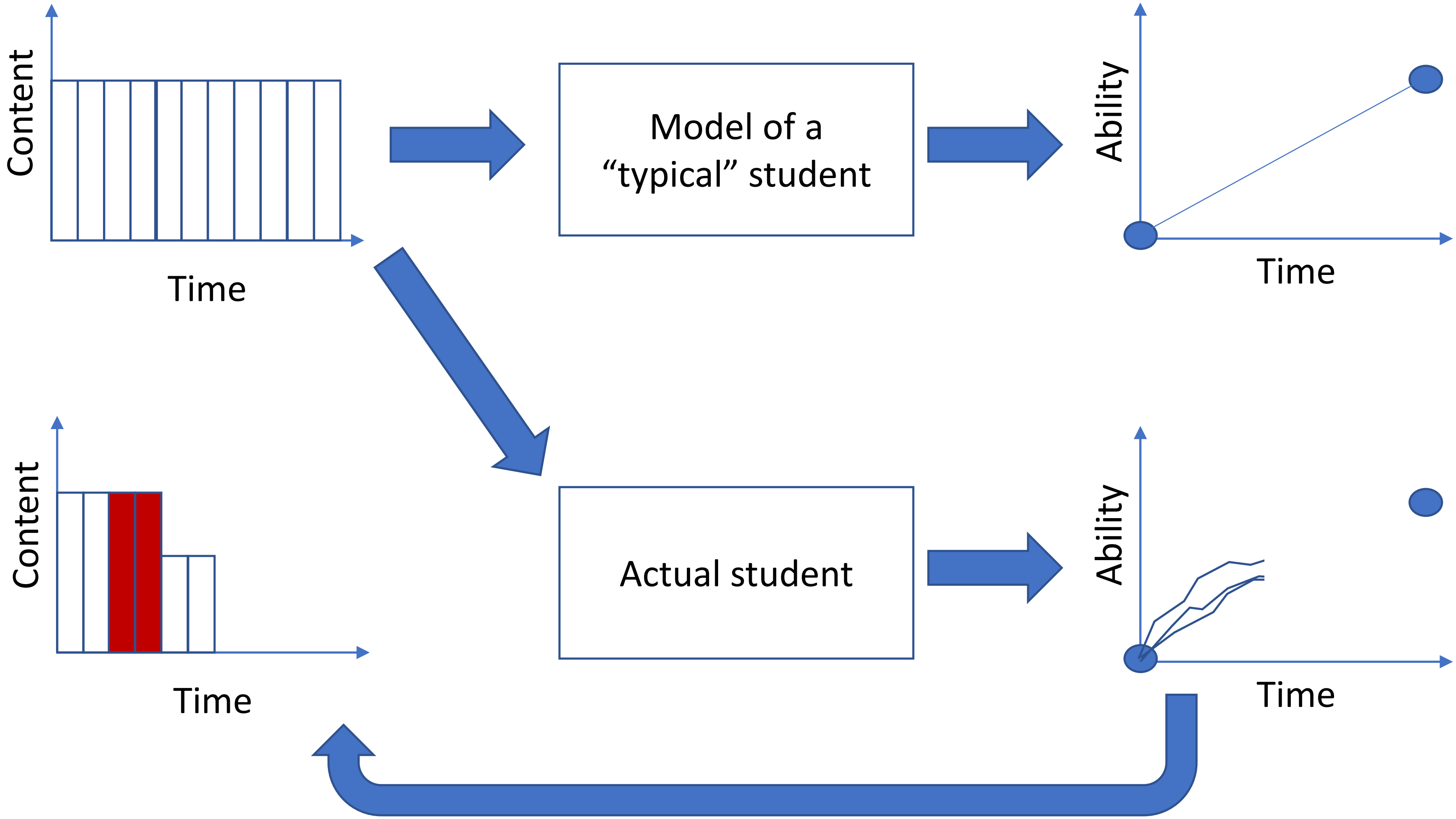
Closed-loop control



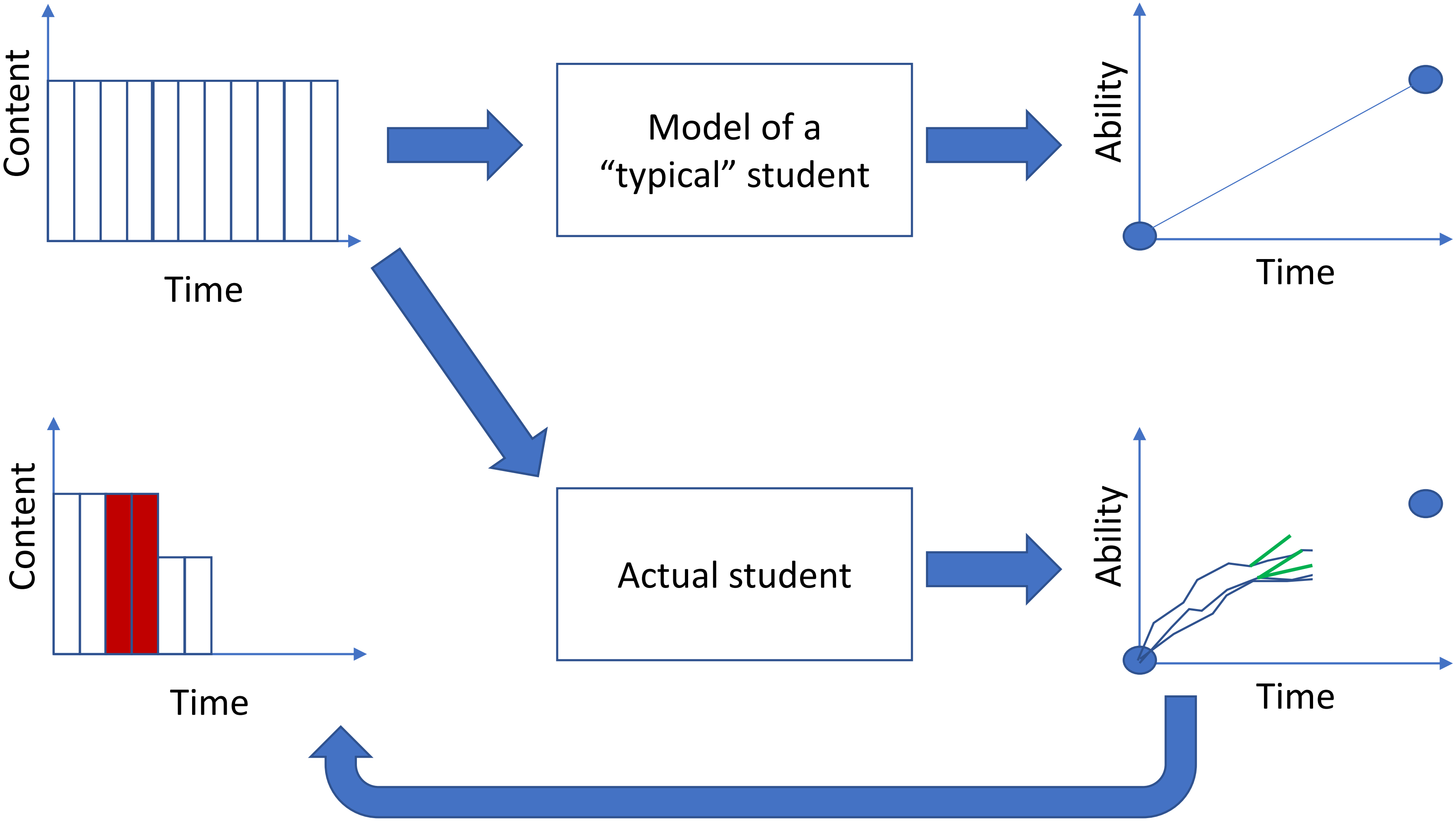
Model of a "typical" student

Actual student

Closed-loop control



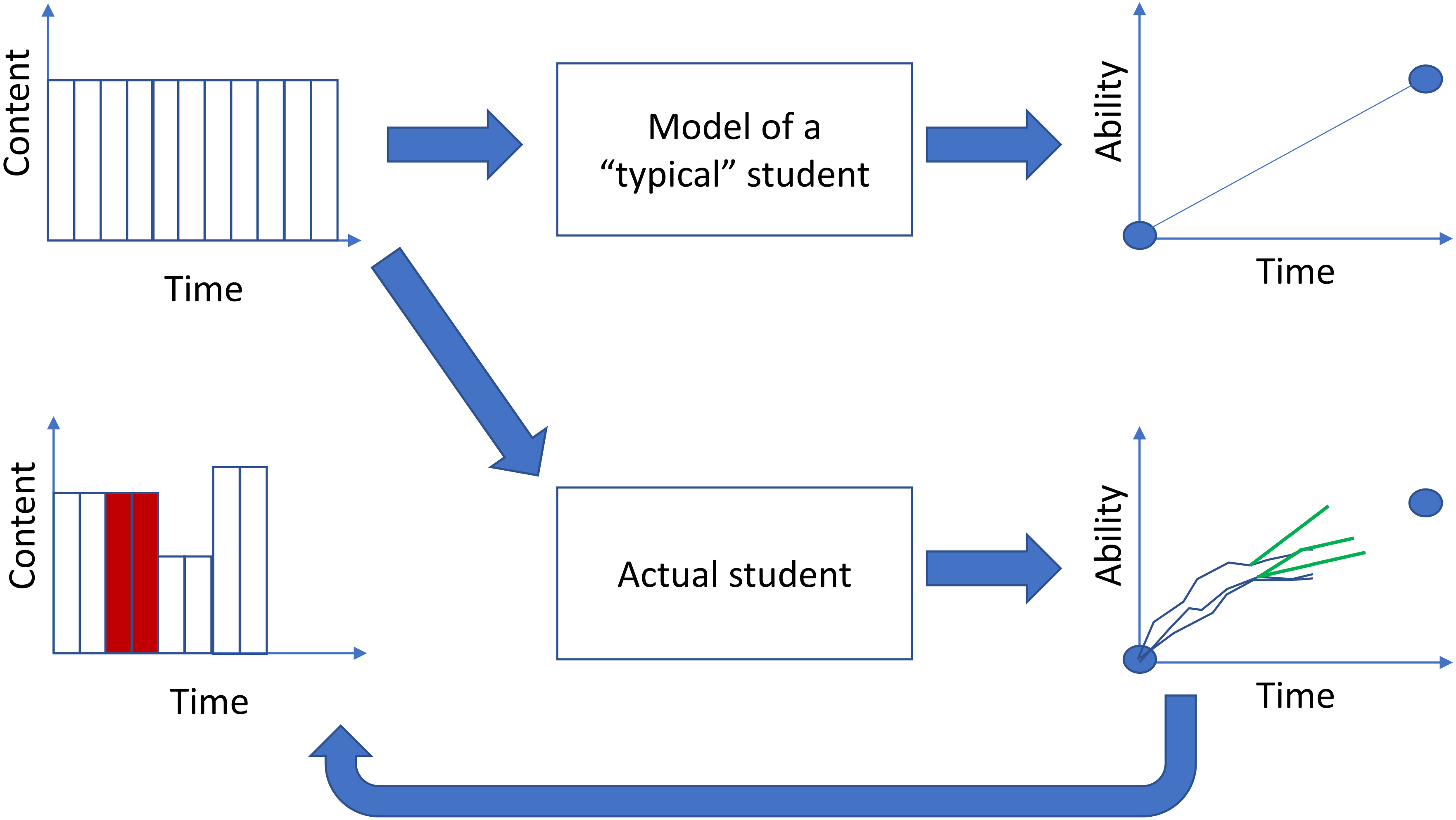
Closed-loop control



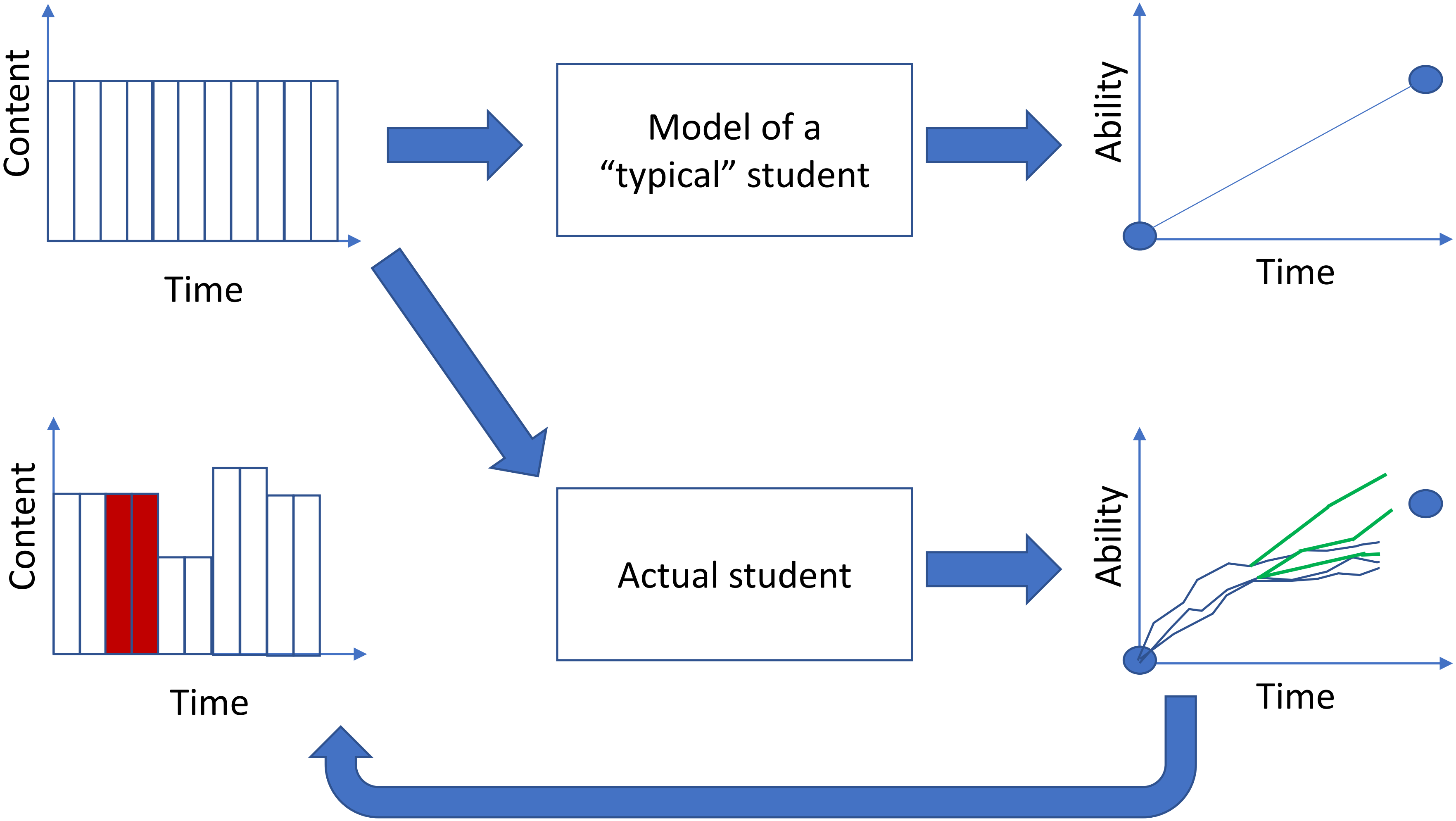
Model of a “typical” student

Actual student

Closed-loop control



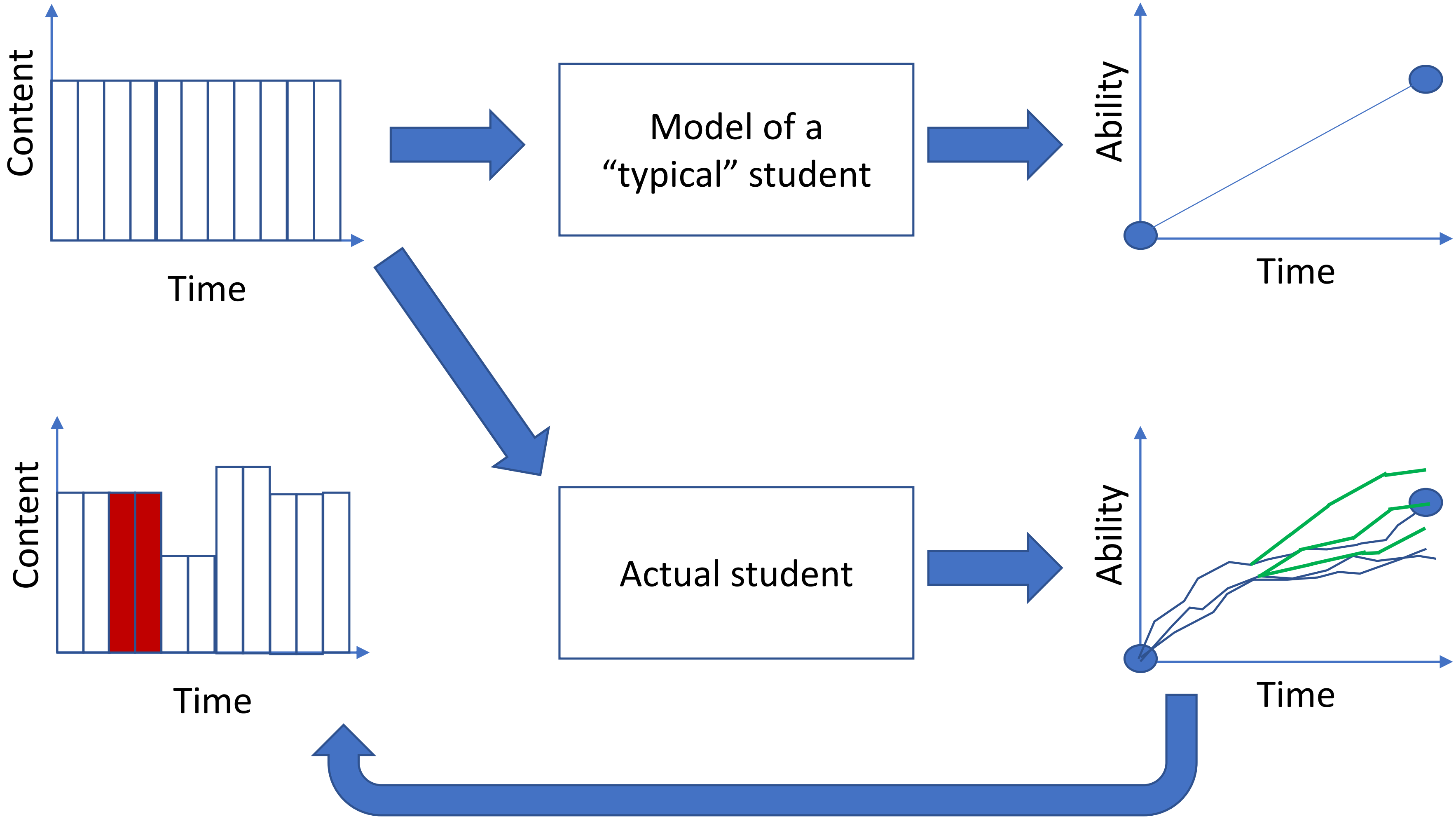
Closed-loop control



Model of a “typical” student

Actual student

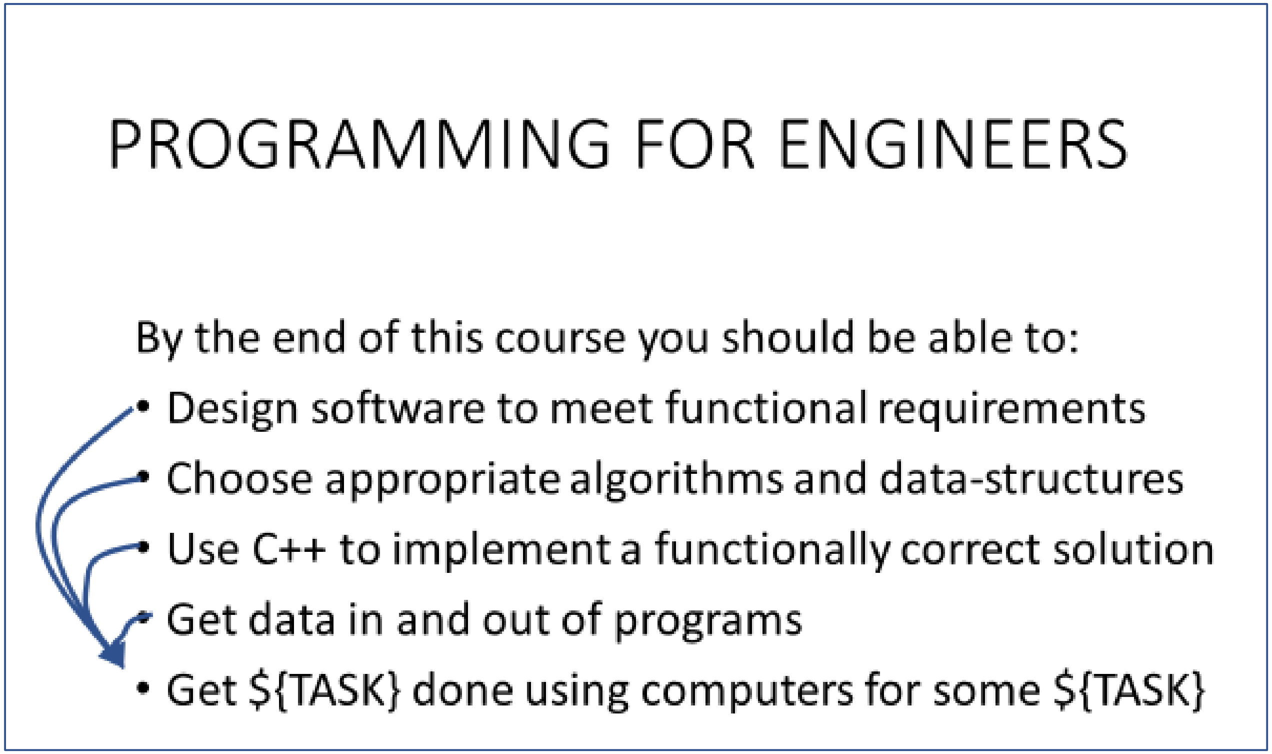
Closed-loop control



Pros and cons of closed-loop

- *Positives*: it is much more accurate
  - Far less likely to over-load or under-load
  - Much more likely to achieve the desired goals
  - Avoids over-corrections if goal will be missed

- *Negatives*: much more expensive
  - Have to feed back from output to input in time
  - Can’t start creating the next input till previous output
  - Effort cannot be scheduled in advance
  - More difficult to finely tune the input



PROGRAMMING FOR ENGINEERS

By the end of this course you should be able to:

- Design software to meet functional requirements
- Choose appropriate algorithms and data-structures
- Use C++ to implement a functionally correct solution
- Get data in and out of programs
- Get ${TASK} done using computers for some ${TASK}

- Our objectives are clear and fixed. But:
  - We know this is ambitious in terms of content
  - We are using new techniques and methods
  - It isn’t clear how students will react

Our approach: open + closed-loop[1]

- Make an initial open-loop plan
  - Develop all the lectures, exercises, labs
  - Get them to the point that they could be delivered
- Every week Sahbi and I discuss the previous week
  - How were lectures, labs?
  - What parts were easy or hard?
  - What questions are being asked?
- Refine as we go in an open-loop way
  - Some lectures are almost exactly as planned
  - Other lectures completely re-written during the week
  - Most portfolio exercises written 6 months ago
  - Some exercises substituted one day before release

[1] - Technically this is called “model-predictive control” in control-theory terms

The objective

We want you to solve problems using software

So the final assessment is:

Solve problems using software

We know exactly what those problems are already

our job is to make sure you are prepared

Solving problems

So far you have done exercises

- Lab exercises are sub-tasks within programming
  - Solving a local problem
  - Often there is only loosely a real “problem”

- Portfolio exercises are mostly micro-tests
  - Any complexity is because you are still learning
  - Most should be relatively straightforward after TBL+labs

- Portfolio + lab are about learning
  - If they take longer, you need more time to ***learn*** the content

The final assessment is “real”

What you get:

- A specification for a particular problem domain
- A set of deliverables associated with that domain

What we are assessing:

- Can you create a solution to a meaningful problem?
- Do you know how to apply engineering techniques?
- Do you have enough coding ability to implement it?

There is ***no*** competition

- All assessment is against fixed criterion
    - There is no competitive aspect
    - No grading on the curve
    - You are not competing against other students
- This is ***not*** a zero-sum game
    - Everyone could do really well
    - Everyone could do really poorly
- Any moderation applies to the entire cohort
    - Only kicks in if the module is much too easy or hard
    - Standard formal process used for exams

Summative vs formative assessment

- Formative assessment:
  - Tells ***you*** how well you are *learning*
- Summative assessment
  - Tells ***others*** how well you *learnt* it : are you good at it?

- Portfolio: completely formative*; work hard -> get marks*
- MCQ Tests: mostly formative; *study material -> do well*
- Final coursework: *mostly summative*
  - Study alone will get mediocre marks
  - Study plus thinking gets ok marks
  - Study plus thinking plus ability gets good marks

Managing risk

- Software is broken into deliverables
  - Libraries, programs, source files, documentation
- We use this to manage assessment risk
  - You will be given a clear list of deliverables
  - These are similar to multiple question parts in an exam
- Designed to be independent
  1. Create a function that...
  2. Create a script that uses existing program to ...
  3. Create a test-case that...
  4. Write a program that...
  5. Add a feature to ...

How should we practice?

- Labs + portfolio are already preparing you
  - What we have done yet is bigger problems
- We have written three final assessments
  - Two will be released in Spring term for practise
  - One will be the final assessment
- *But*: they rely on things you haven’t seen yet
  - Things like OOP are needed
  - Need tools that will be used for pair work

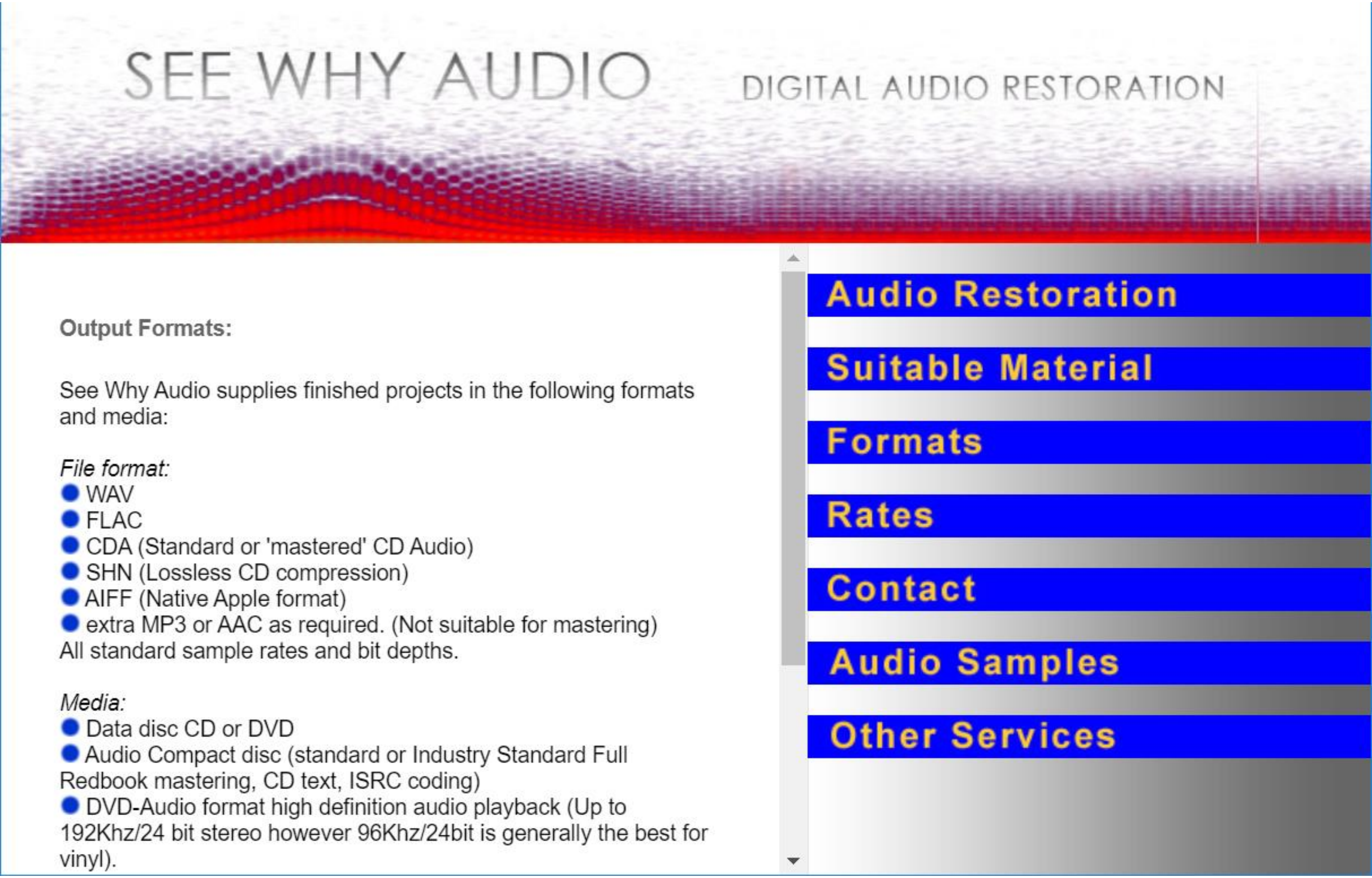
A question for you

- The coursework is done in pairs
  - Collaboration is an important skill
  - One person can’t get enough done in one day
- You need to be assigned pairs ahead of time
  - You’ll need to practice working with your partner
  - Pairs need to be known by Spring mid-term
- We have two choices for pair assignment
  1. Random assignment: *fair and realistic*
  2. Self assignment: *often causes less friction*
- Currently we are planning random
  - *Does the cohort have strong opinions on this?*

Practise problems

- We have a set of reduce problems to release at the end of term
  - You can look at them over Christmas; or not.
- These are ***not*** assessed in any way
  - Merely practise problems to try solving
- They are ***not*** complete assessments
  - But they allow you to apply current programming skills
  - They also model the types of activity needed

Colin and his records



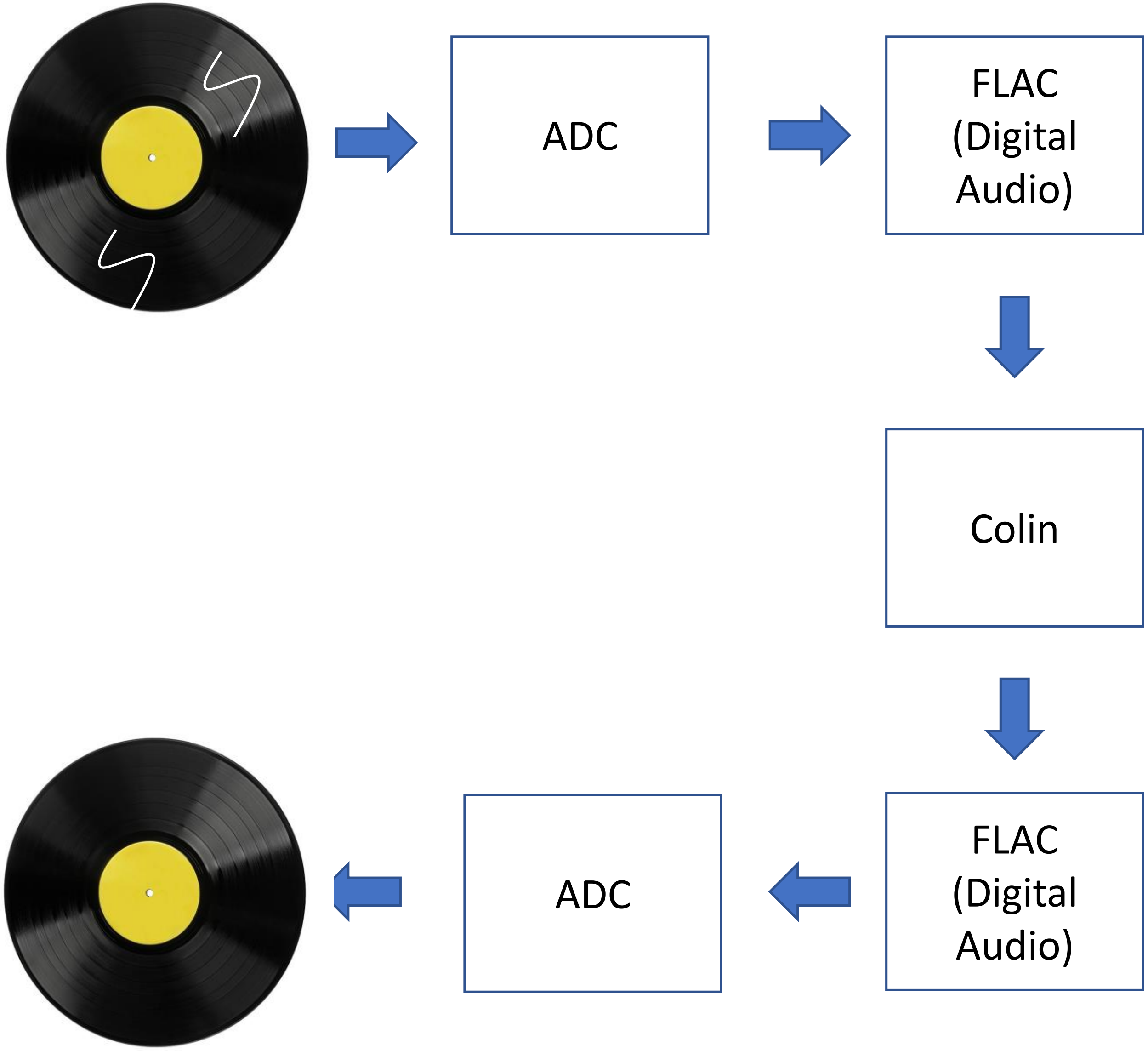
Output Formats:

See Why Audio supplies finished projects in the following formats and media:

File format:
- WAV
- FLAC
- CDA (Standard or 'mastered' CD Audio)
- SHN (Lossless CD compression)
- AIFF (Native Apple format)
- extra MP3 or AAC as required. (Not suitable for mastering)

All standard sample rates and bit depths.

Media:
- Data disc CD or DVD
- Audio Compact disc (standard or Industry Standard Full Redbook mastering, CD text, ISRC coding)
- DVD-Audio format high definition audio playback (Up to 192Khz/24 bit stereo however 96Khz/24bit is generally the best for vinyl).



ADC

FLAC (Digital Audio)

Colin

FLAC (Digital Audio)

ADC

The problem

Example deliverables

1. Create a function that finds the difference between two samples
2. Write a C++ program to calculate a histogram of delayed samples
3. Write a C++ program to delete repeated audio samples in raw audio
4. Create a script that uses that program to remove repeats in flac files
5. Write a C++ program that applies a custom distance metric to detect repeats
6. Write a program that uses the inverse of numeric differentiation to estimate near-repeat frequency

Finishing off

- Portfolio
  - Sanity-test on Monday morning
  - Final due next Friday
- January test
  - Expected to be Monday morning on the 6th
- Example problems
  - Released today or tomorrow via blackboard

Have a good Christmas (or equivalent)