

CS 361

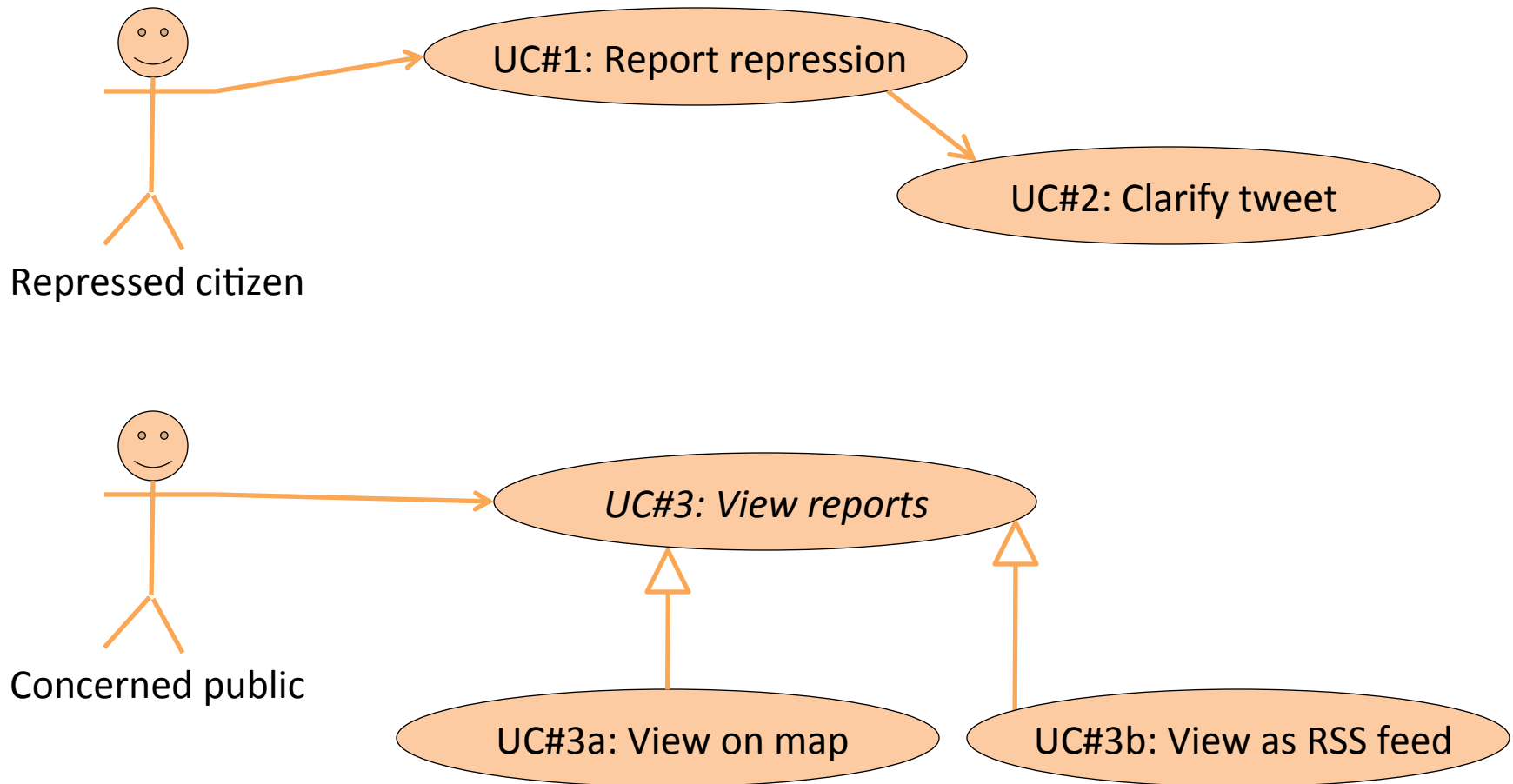
Software Engineering I

Effort

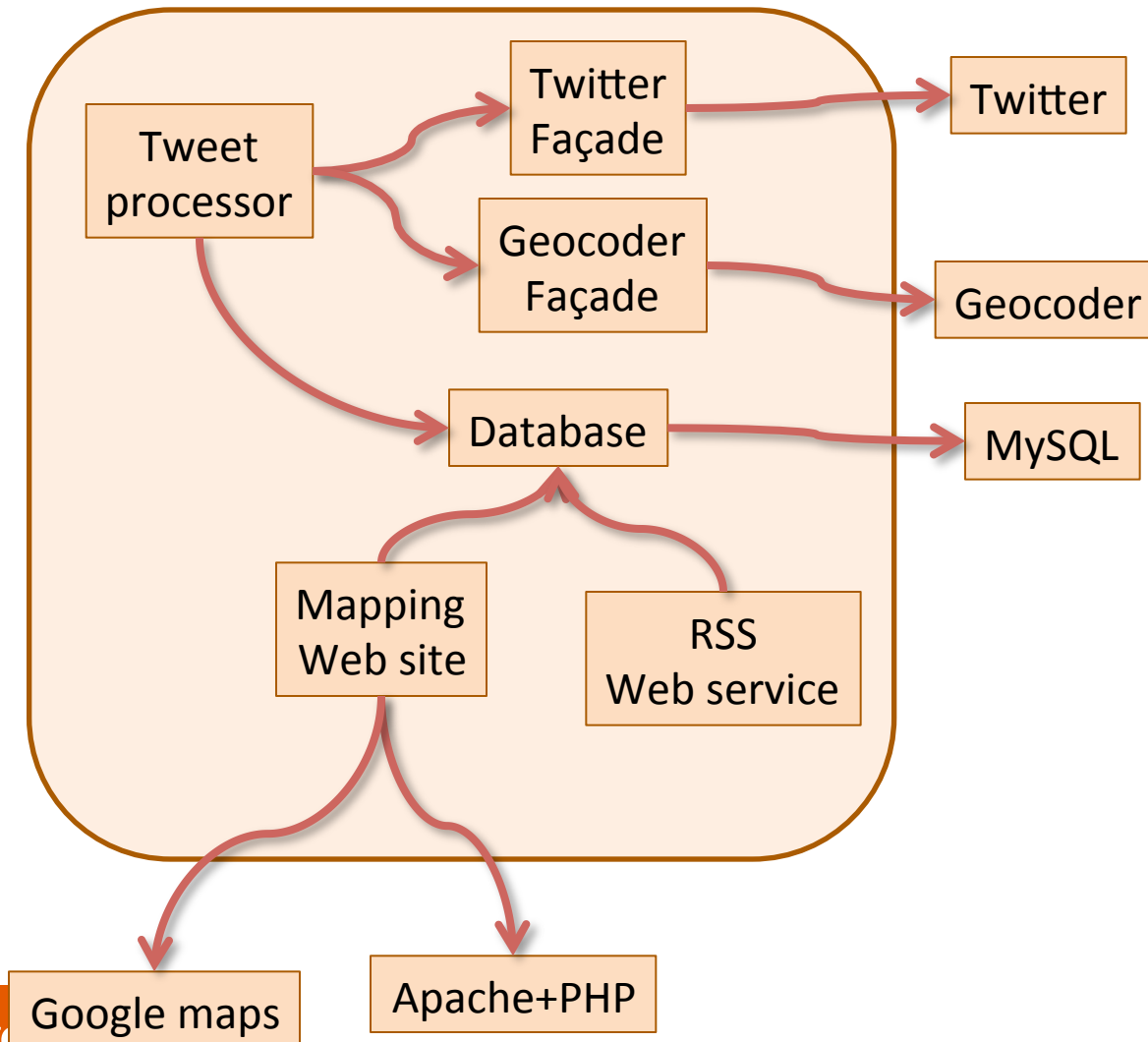
Planning big projects

1. Figure out what the project entails
 - Requirements, architecture, design
2. Figure out **dependencies & priorities**
 - What has to be done in what order?
3. Figure out **how much effort** it will take
4. **Plan, refine, plan, refine, ...**

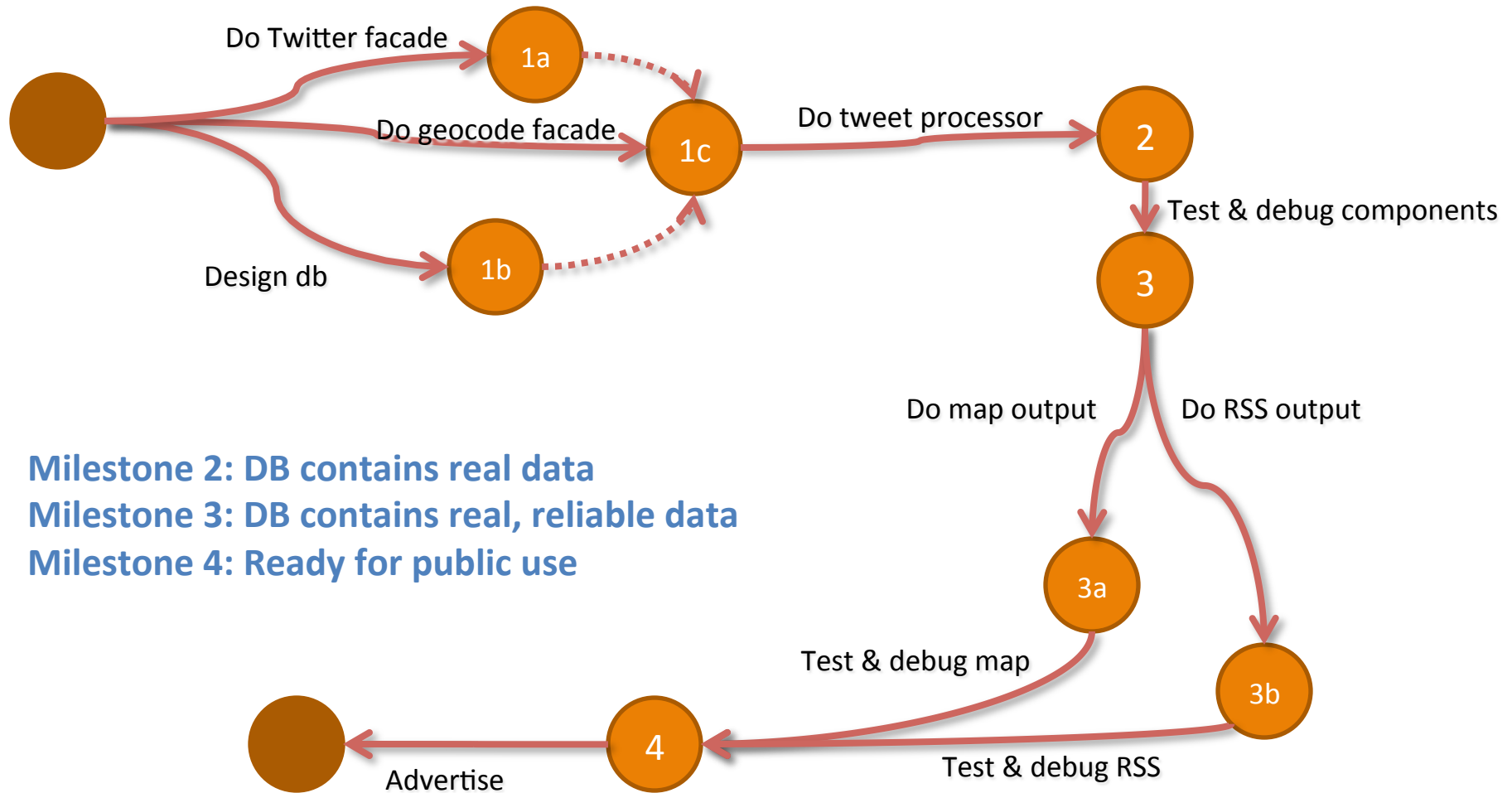
Example: Twitter repression report



One possible architecture



Activity graph: shows dependencies of a project's activities



Activity graph: shows dependencies of a project's activities

- Filled circles for start and finish
- One circle for each milestone
- Labeled arrows indicate activities
 - What activity must be performed to get to a milestone?
 - Dashed arrows indicate “null” activities



Effort

- Ways to figure out effort for activities
 - Expert judgment
 - Records of similar tasks
 - Effort-estimation models
 - Any combination of the above

Expect to refine effort estimates

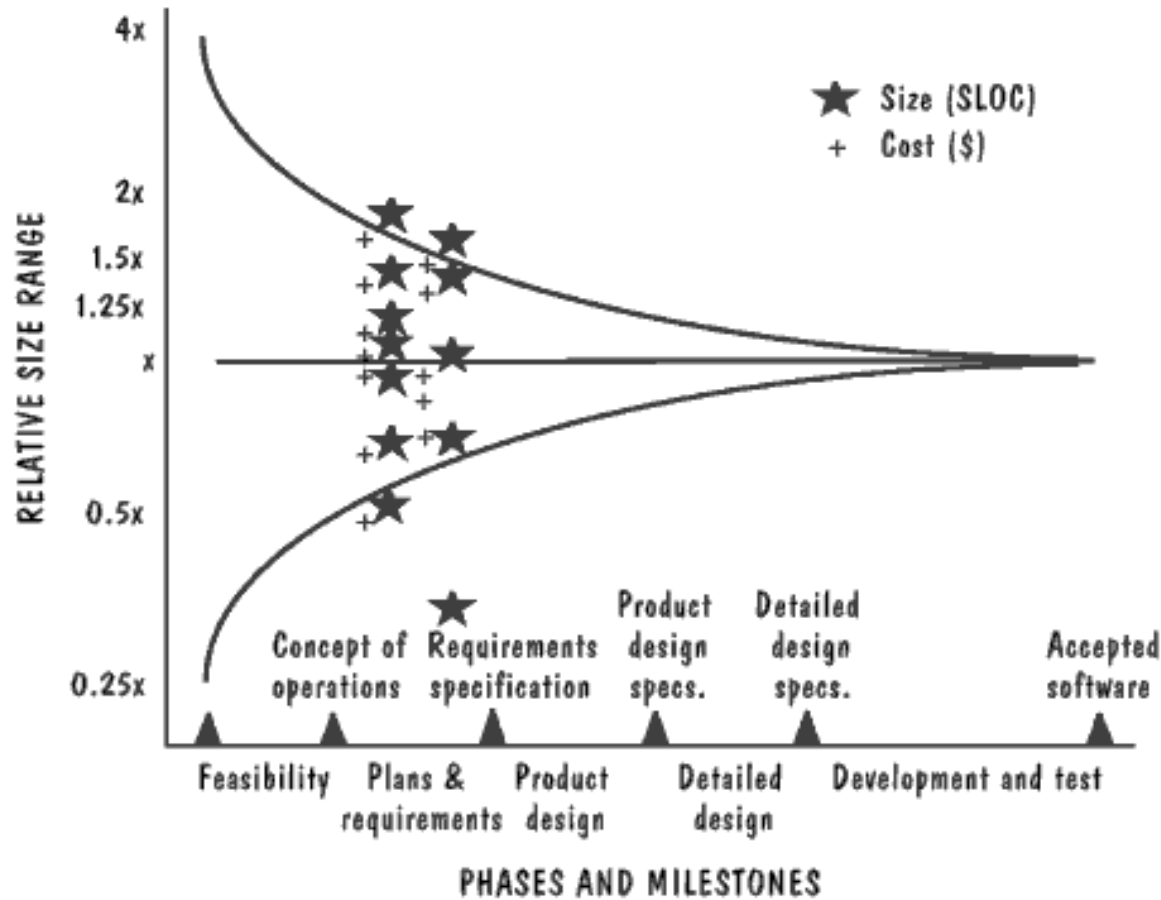


FIGURE 3.12 Changes in estimation accuracy as project progresses (Boehm et al. 1995).



Effort: expert judgment

- Not a terrible way to make estimates, but...
 - Often vary widely
 - Often wrong
 - Can be improved through **iteration & discussion**
- How long to do the following tasks:
 - Read tweets from Twitter via API?
 - Send tweets to Twitter via API?
 - Generate reports with Google maps?

Effort: records of similar tasks

- **Personal software process (PSP)**
 - Record the size of a component (lines of code)
 - Breakdown # of lines added, reused, modified, deleted
 - Record time taken
 - Breakdown planning, design, implement, test, ...
 - Refer to this data when making future predictions
- Can also be done at the team level



Effort: estimation models

- Algorithmic (e.g.: COCOMO)
 - Inputs = description of project + team
 - Outputs = estimate of effort required
- Machine learning (e.g.: CBR)
 - Gather descriptions of old projects + time taken
 - Run a program that creates a model
 - ➔ You now have a custom algorithmic method
 - Same inputs/outputs as algorithmic estimation method



Using COCOMO-like models

1. Assess the system's complexity
2. Compute the # of application points
3. Assess the team's productivity
4. Compute the effort



Assessing complexity

TABLE 3.10 Application Point Complexity Levels

For Screens				For Reports			
	Number and source of data tables				Number and source of data tables		
<i>Number of views contained</i>	Total < 4 (<2 servers, <3 clients)	Total < 8 (2–3 servers, 3–5 clients)	Total 8+ (>3 servers, >5 clients)	<i>Number of sections contained</i>	Total < 4 (<2 servers, <3 clients)	Total < 8 (2–3 servers, 3–5 clients)	Total 8+ (>3 servers, >5 clients)
<3	Simple	Simple	Medium	0 or 1	Simple	Simple	Medium
3–7	Simple	Medium	Difficult	2 or 3	Simple	Medium	Difficult
8+	Medium	Difficult	Difficult	4+	Medium	Difficult	Difficult

e.g.: A screen for editing the database involves 6 database tables, and it has 4 views. This would be a “medium complexity screen”.

This assessment calls for *lots* of judgment.



Computing application points (a.p.)

TABLE 3.11 Complexity Weights for Application Points

Element Type	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component	—	—	10

e.g.: A medium complexity screen costs 2 application points.

3GL component = reusable programmatic component that you create



Assessing team capabilities

TABLE 3.12 Productivity Estimate Calculation

Developers' experience and capability	Very low	Low	Nominal	High	Very high
CASE maturity and capability	Very low	Low	Nominal	High	Very high
Productivity factor	4	7	13	25	50

- *extra low* if it has fewer than 3 months of experience
- *very low* if it has at least 3 but fewer than 5 months of experience
- *low* if it has at least 5 but fewer than 9 months of experience
- *nominal* if it has at least 9 months but less than one year of experience
- *high* if it has at least 1 year but fewer than 2 years of experience
- *very high* if it has at least 2 years but fewer than 4 years of experience
- *extra high* if it has at least 4 years of experience

TABLE 3.13 Tool Use Categories

Category	Meaning
Very low	Edit, code, debug
Low	Simple front-end, back-end CASE, little integration
Nominal	Basic life-cycle tools, moderately integrated
High	Strong, mature life-cycle tools, moderately integrated
Very high	Strong, mature, proactive life-cycle tools, well-integrated with processes, methods, reuse

e.g.: Productivity with low experience + nominal CASE... productivity = $(7+13)/2 = 10$

Application points per person-month (assuming NO vacation or weekends!!!!)



A word about CASE tools

- “Some typical CASE tools are:
 - Configuration management tools
 - Data modeling tools
 - Model transformation tools
 - Program transformation tools
 - Refactoring tools
 - Source code generation tools, and
 - Unified Modeling Language”

– Wikipedia

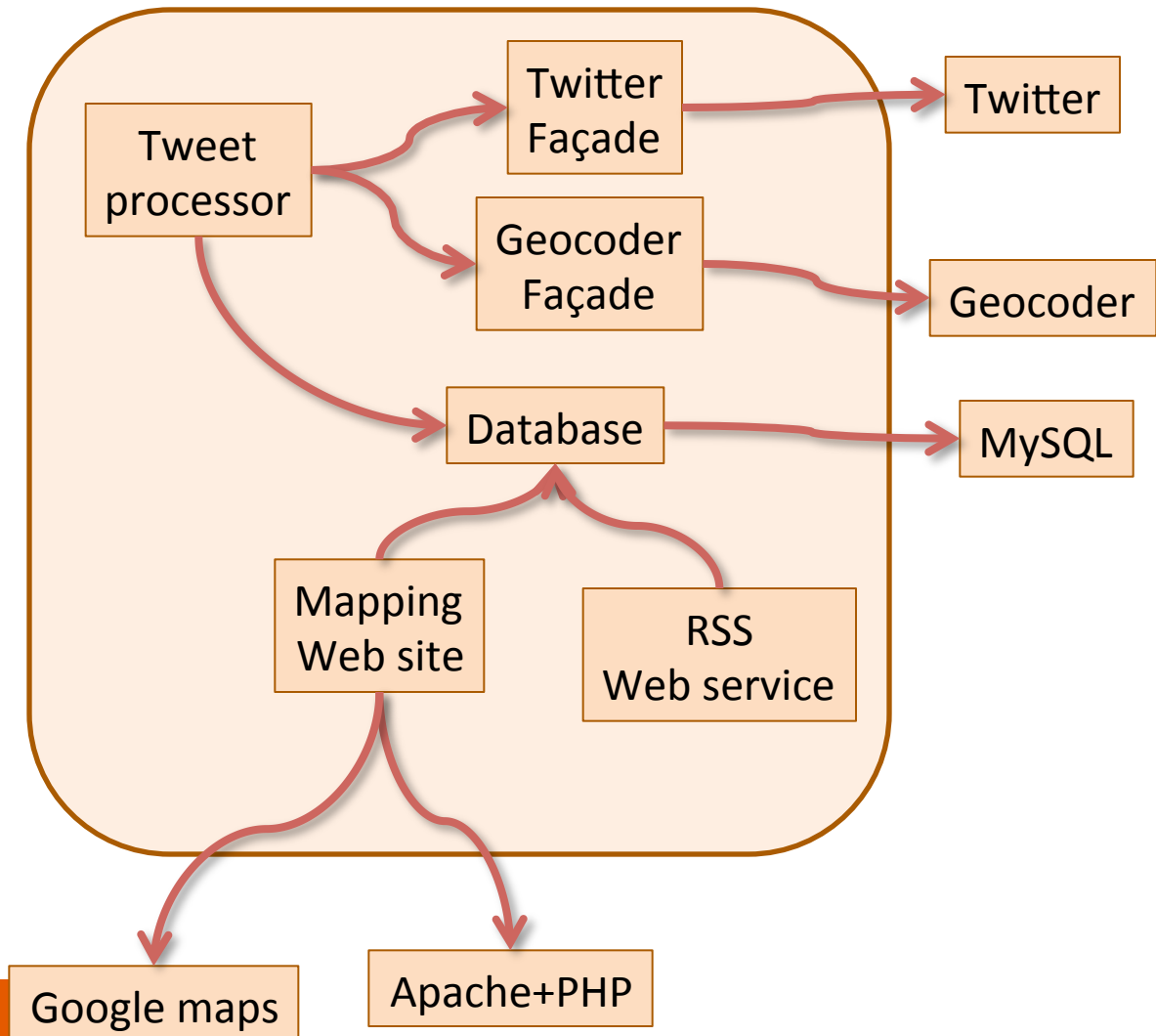
Identify screens, reports, components

3GL components

- Tweet processor
- Twitter façade
- Geocoder façade

Reports

- Mapping web site
- RSS web service

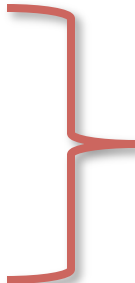


Use complexity to compute application points

3GL components

- Tweet processor
- Twitter façade
- Geocoder façade

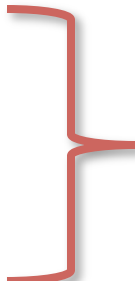
Simple model assumes that all 3GL components are 10 application points.


$$3 * 10 = 30 \text{ a.p.}$$

Reports

- Mapping web site
- RSS web service

Displays data from only a few database tables (3? 4?)
Neither has multiple sections.
→ Each is probably a “simple” report, 2 application points.

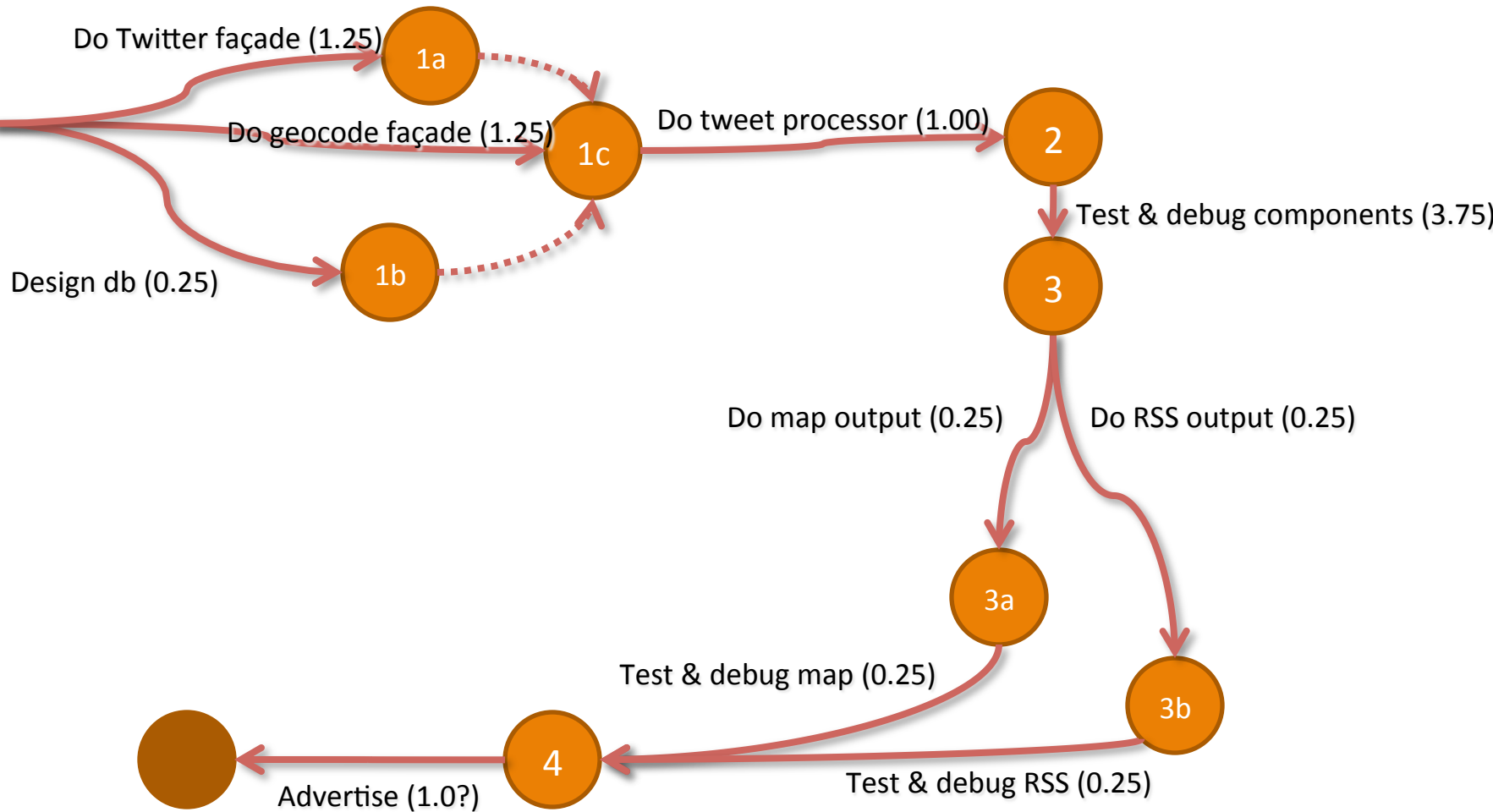

$$2 * 2 = 4 \text{ a.p.}$$

$$30 + 4 = 34 \text{ a.p.}$$

Assess the team's productivity & compute effort

- At one company where I worked...
 - Extensive experience with websites, XML
 - But no experience with Twitter or geocoders
 - Since 20 of the 34 a.p. are on this new stuff, assume very low experience
 - Virtually no CASE support... very low
- ➔ productivity = $(4 + 4) / 2 = 4$ a.p. / month
- So 34 a.p. would take **8.5 person-months**
- Note: this assumes no vacation or weekends

Distribute the person-months over the activity graph



The magic behind distributing person-months

- Divide person-months between implementation and other activities (design, testing, debugging)
 - Oops, forgot to include an activity for testing and debugging the components... revise activity graph
- Notice that some activities aren't covered
 - E.g.: advertising; either remove from diagram or use other methods of estimation

Do you believe those numbers?

- Ways to get more accurate numbers:
 - Revise numbers based on expert judgment or PSP
 - Perform a “spike” ... *try* something out and actually *see* how long it takes
 - Use more sophisticated models to analyze how long components will really take
 - Use several models and compare
- Expect to revise estimates as project proceeds



Further analysis may give revised estimates...

