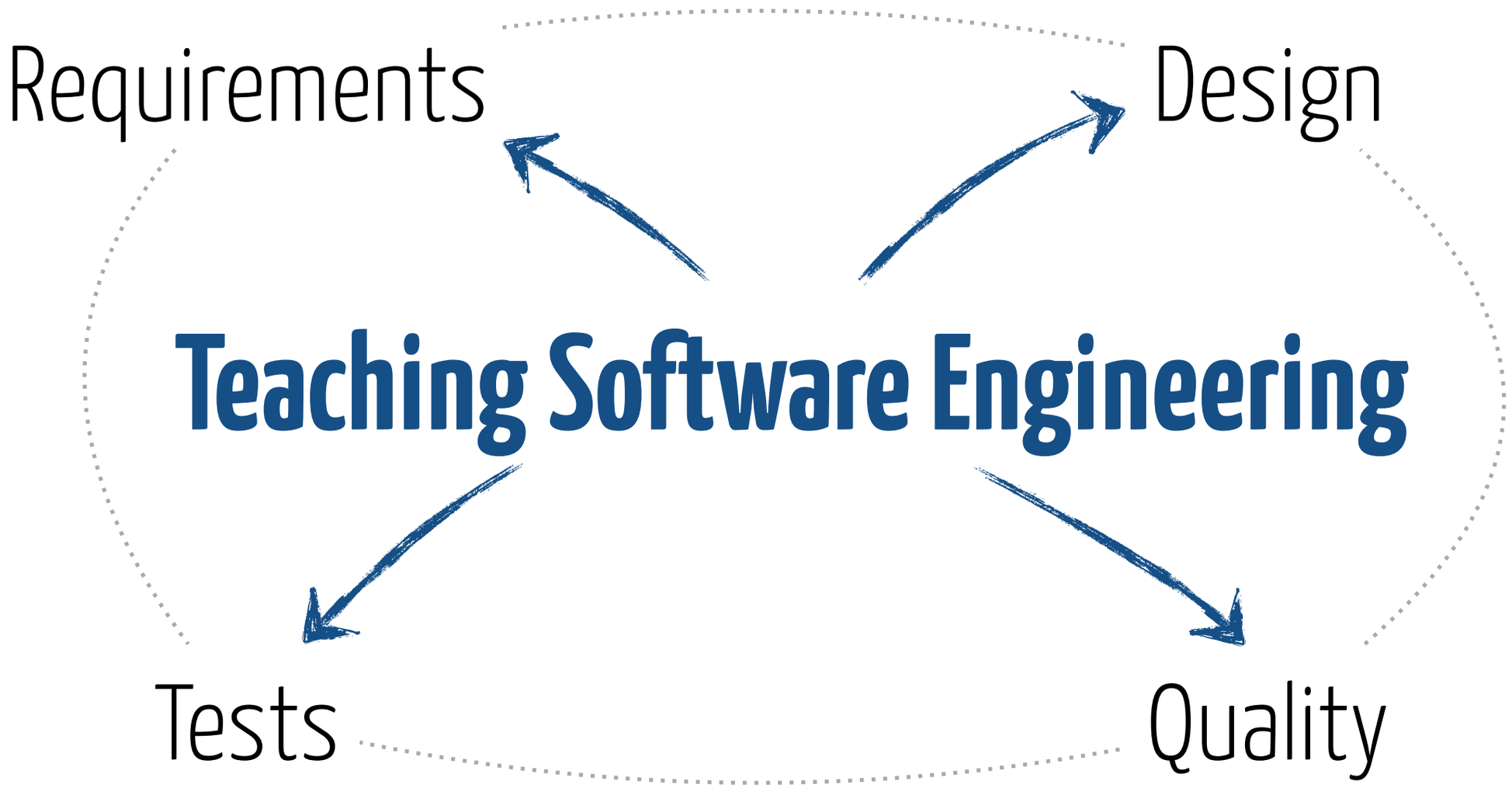
Using “Island” to teach software engineering

Sébastien Mosser
SE@MTL #2, 06.06.2019
mosser.sebastien@uqam.ca

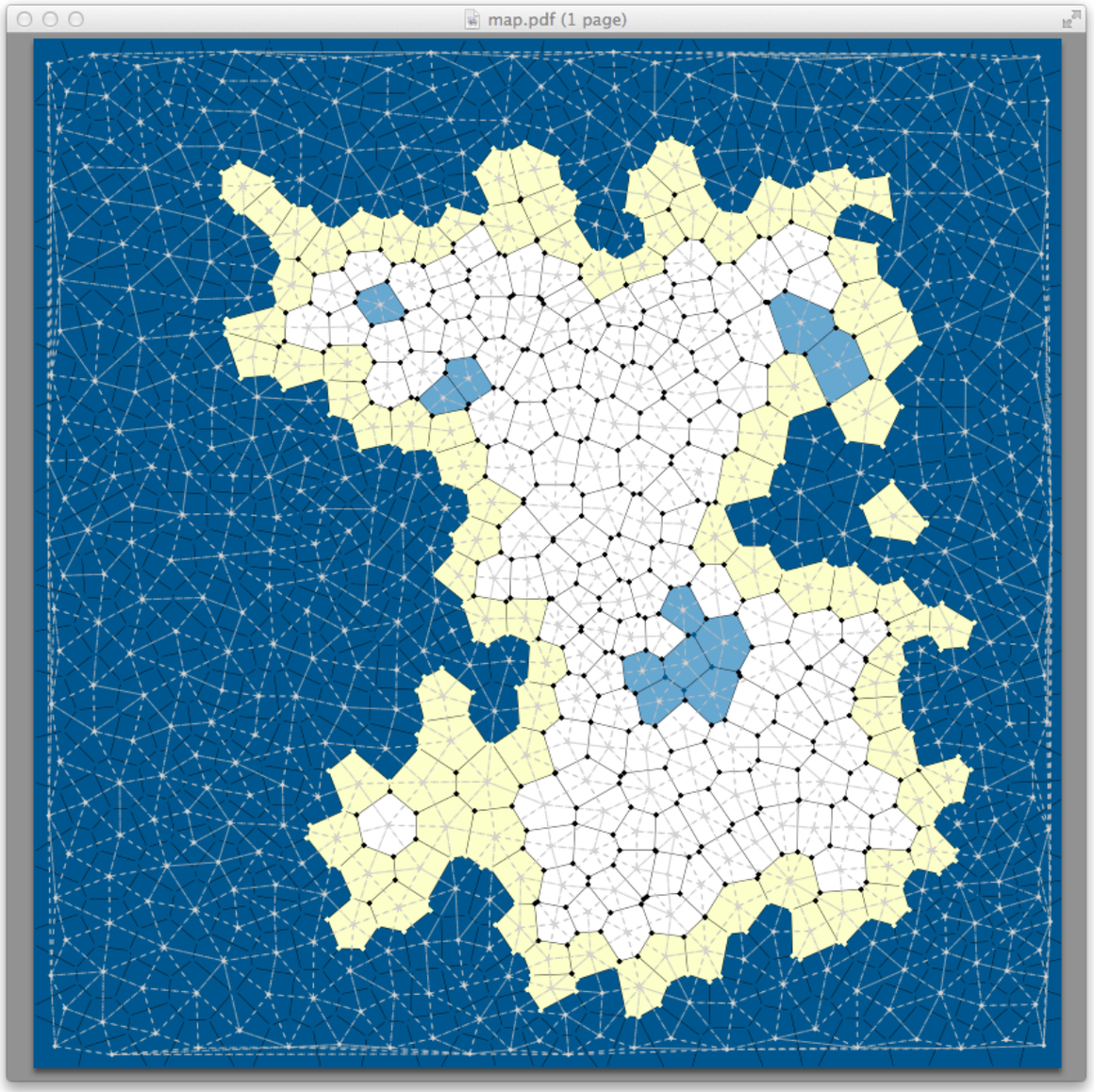
UQÀM

Images: PixaBay

Undergrad curriculum

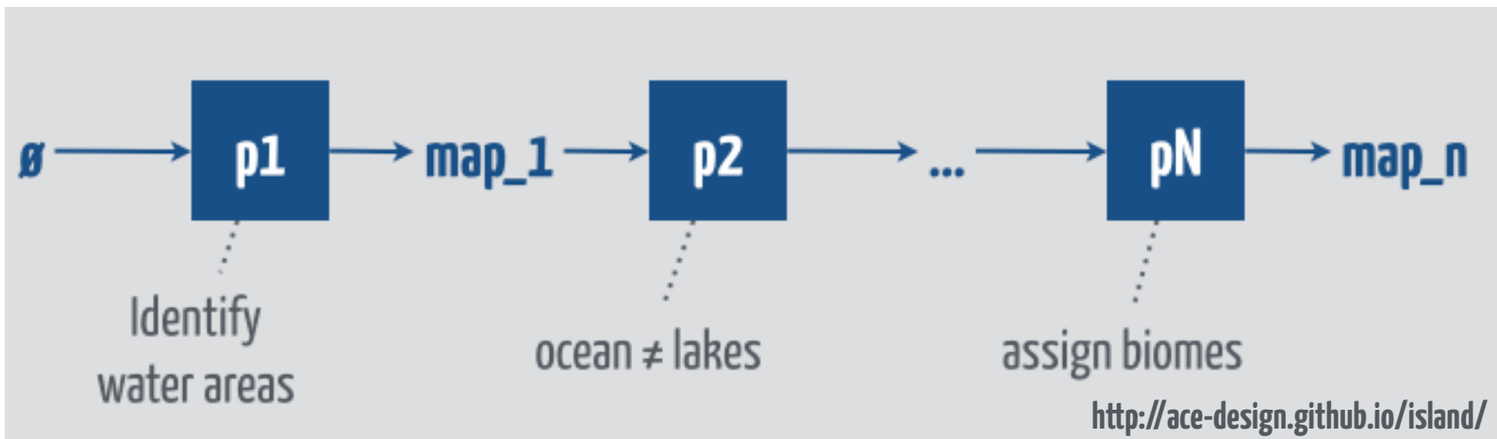


Challenge: Make SE "**fun**" and "**necessary**"

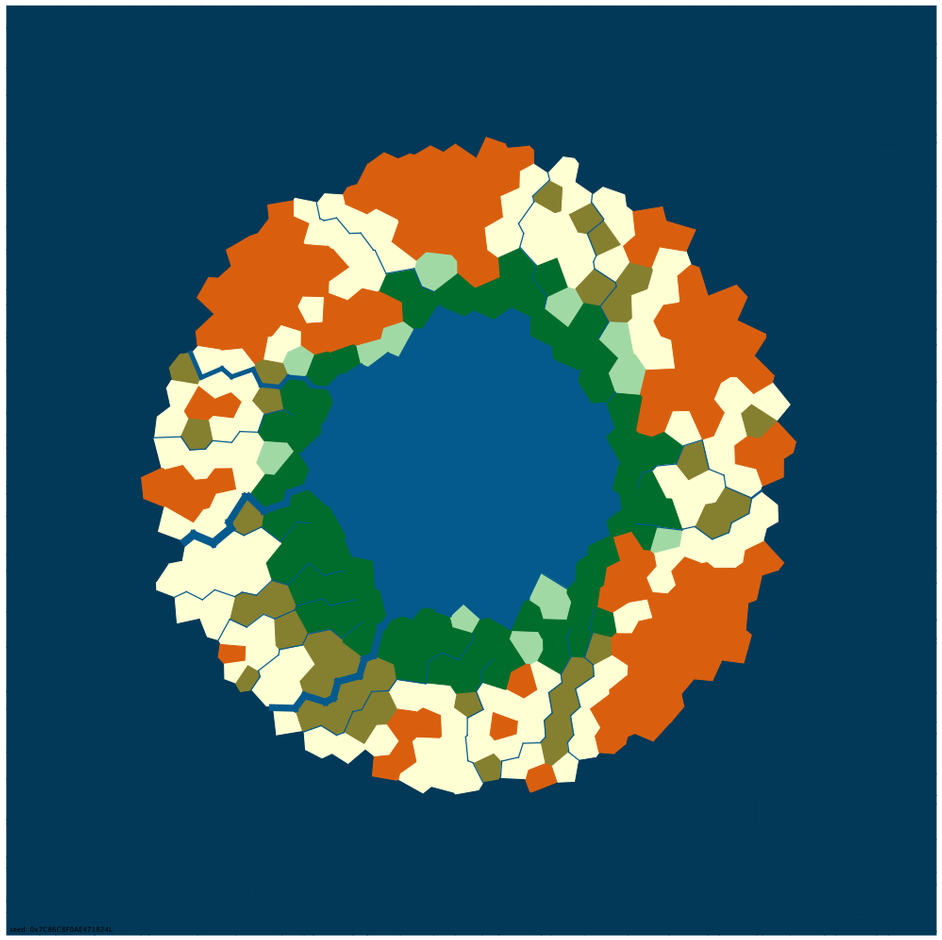


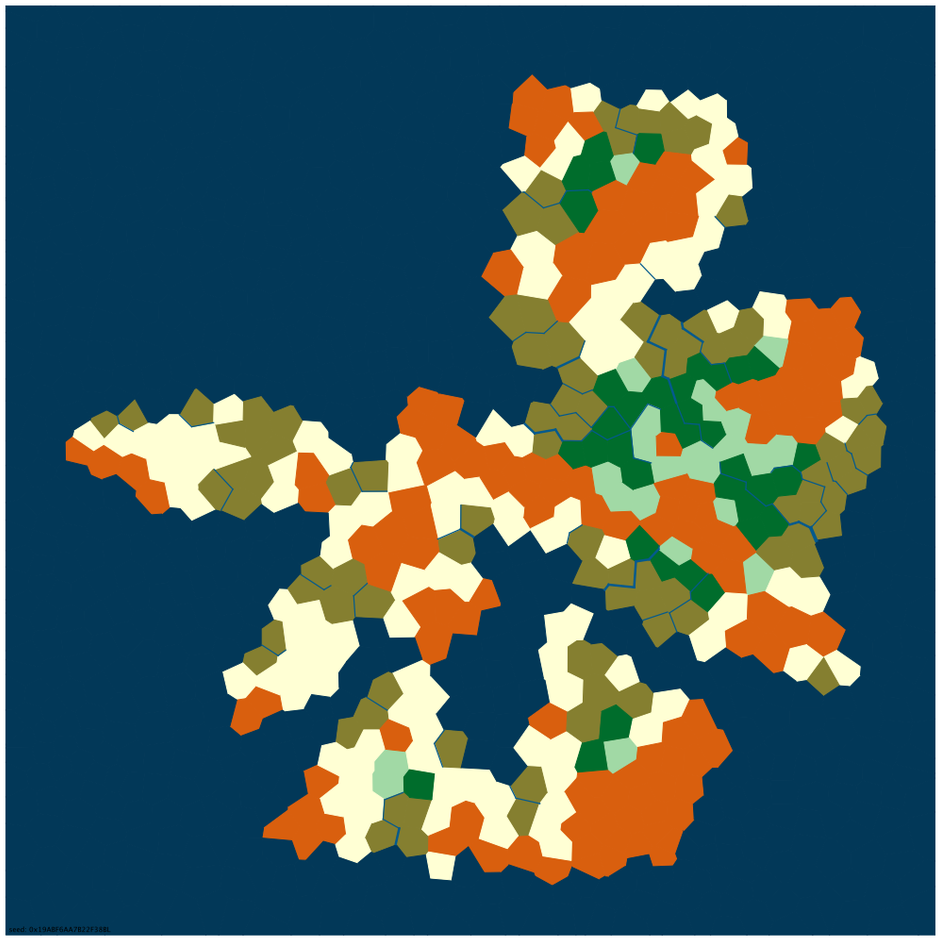
Procedural Terrain Generator

Voronoi diagrams
Delaunay triangulation
Function composition
Whittaker biome model



```
// Round island, quite big. Easy to exploit.
val s47 = 0x7C86C8F0AE471824L
lazy val week47: IslandMap = {
  createIsland shapedAs donut(70.percent, 30.percent) withSize 1600 having 1200.faces builtWith Seq(
    plateau(30), flowing(rivers = 30, distance = 0.4), withMoisture(soils.normal, distance = 700),
    AssignPitch, usingBiomes(WhittakerDiagrams.caribbean)) usingSeed s47
}
```





```
// Needle in an haystack
val s49 = 0x19ABF6AA7B22F38BL
lazy val week49: IslandMap = {
  createIsland shapedAs radial(factor = 1.57) withSize 1600 having 1200.faces builtWith Seq(
    plateau(30), flowing(rivers = 40, distance = 0.1), withMoisture(soils.wet, distance = 100),
    AssignPitch, usingBiomes(WhittakerDiagrams.caribbean)) usingSeed s49
}
```

Gamification: Exploiting Island resources

```
IExplorerRaid raid = new MyExplorer();

String context = "{ ... }";
raid.initialize(context);

while ( !endOfGame ) {
  String decision = raid.takeDecision();
  String result  = engine.compute(decision);
  raid.acknowledgeResults(result);
}

String report = raid.deliverFinalReport();
```

The bot is asked to
(i) find the island, and then
(ii) collect resources

Action	Phase	Cost Cat.
FLY	1	cheap
HEADING	1	medium
ECHO	1	cheap
SCAN	1	medium
STOP	1 & 2	variable
LAND	1 & 2	expensive
MOVE_TO	2	variable
SCOUT	2	medium
GLIMPSE	2	cheap
EXPLORE	2	expensive
EXPLOIT	2	expensive
TRANSFORM	2	medium

Championship: deliver value each week

	Sandbox (creeks finding)						Production (contracts harvesting)											Points			Rétrospectives	
Team	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	*Sand*	*Prod*	*Total*	15	20
E			1								3	7	2	2	5	5	2	*3*	*18,5*	***21,5***	*39*	65
G						3			2		7	3	3	1	4	2	3	*3,5*	*23*	***26,5***	*42*	65
I					3	6				3	1	1	1		1		1	*4*	*25,5*	***29,5***	*43*	64
C				3	4	4			3	2		6	10	3	3	3	11	*5,5*	*16,5*	***22***	*16*	61
O					6	2			1	1	6	5	4	5		10	7	*0*	*19,5*	***19,5***	*41*	61
L				1		1			5		5	2	7	7		1	10	*14*	*16,5*	***30,5***	*40*	60
A				4	5				7		2		9	9	9	4	12	*0*	*11*	***11***	*28*	55
H					1						11	10	5	6	2	6	4	*0*	*12*	***12***	*30*	54
F						5					4	4	8		7	8	5	*0*	*8,5*	***8,5***	*23*	53
D				2	2			1	8	4	9		11		6	7	8	*9,5*	*14,5*	***24***	*10*	50
J					7	7							13			13		*2,5*	*2*	***4,5***	*2*	46
N											10	8	6	4	8	9	13	*0*	*9*	***9***	*36*	36
M						8			6		8		14	8	10	11	9	*0*	*3,5*	***3,5***	*0*	34
K									4			9	12		11	12	6	*0*	*0*	***0***	*14*	21
P																15	14	*0*	*0*	***0***	*0*	21
B																14		*0*	*1*	***1***	*2*	0
MVP																		***3***	***5,5***	***8,5***	*0*	0
Champ	1	1	1	1	1	1	1	1	1	1	4	4	4	1	4	6	4	*36*	*36*	***72***	*42*	62

Regularity is evaluated

(Winning the championship is not)

Requirements

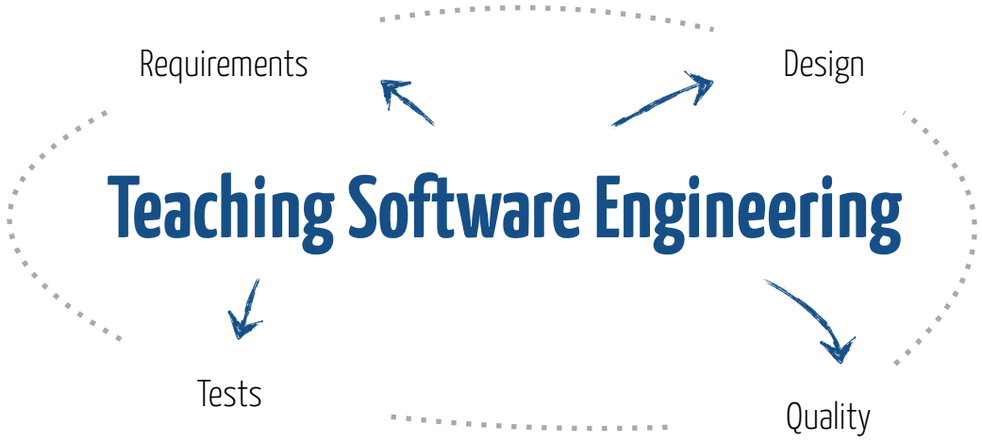
Fuzzy Specification

Feature-driven development

Design

Object-oriented patterns

SOLID + GRASP



Tests

Unit tests

Regression tests

Software Metrics

Legacy code maintenance

Quality